UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Desarrollo de una aplicación multiplataforma para una ONG

Carlos Iniesta Fernández-Pacheco
Tutor: Miguel Ángel Mora

JUNIO 2019

Desarrollo de una aplicación multiplataforma para una ONG

AUTOR: Carlos Iniesta Fernández-Pacheco
TUTOR: Miguel Ángel Mora

Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen (castellano)

Según el último recuento oficial, alrededor de 650 personas duermen a diario en la calles de Madrid. Todas las noches de lunes a jueves, voluntarios de “Solidarios para el Desarrollo” salen a realizar distintas rutas por la ciudad, visitando a estas personas, compartiendo caldo, café y galletas, pero sobretodo realizando una labor de integración en la sociedad, rompiendo los muros que separan a las personas en situación de calle de la gente que anda por ellas, sirviendo de desahogo y apoyo de estas personas estableciendo una relación de igual a igual con ellas.

El objetivo de este Trabajo de Fin de Grado, es el diseño, desarrollo e implantación de una aplicación web progresiva (PWA) llamada Cantarranas. Desde la cual, los voluntarios que realizan las rutas pueden escribir, publicar y consultar informes de las propias rutas. Y además, permita a los trabajadores de la organización explotar los datos de los informes, generando gráficos con ellos, así como llevando un control de los voluntarios en activo para facilitar y optimizar la organización. Por otro lado, servirá como una pequeña red social de la organización, que conecte a los voluntarios de las distintas rutas, con un espacio donde poder compartir contenido entre ellos y comunicarse con los administradores. Además, contará también con un calendario donde poder consultar los eventos, turnos de limpieza y días de cierre del local de comida.

Estas funcionalidades, se han ido definiendo en diversas reuniones con el responsable del programa de personas sin hogar de Solidarios para el Desarrollo, atendiendo a las necesidades del programa y a las peticiones de los voluntarios.

La aplicación se ha desarrollado a nivel Frontend con el framework de JavaScript Vue.js, apoyado en Nuxt.js y la librería de componentes Vuetify. El Backend se ha implementado con Firebase, utilizando su servicio de Hosting así como de Realtime Database, para construir la base de datos en tiempo real.

La última fase de este proyecto ha consistido en la realización de pruebas reales. En las que la aplicación ha sido utilizada más de 60 voluntarios, de cara a que en septiembre de 2019 su uso sea completo por parte de la organización.

Palabras clave

Personas sin Hogar, Solidarios para el Desarrollo, Cantarranas, Aplicación Web Progresiva, JavaScript, Vue, Nuxt, Vuetify, Firebase.

Abstract (English)

According to the last official count, around 650 people sleep daily in the streets of Madrid. Every night from Monday to Thursday, “Solidarios para el Desarrollo” volunteers go out to make different routes around the city, visiting these people, sharing broth, coffee and cookies, but above all doing a job of integration in society, breaking the walls that separate the people living on the streets from the people who walk by them, serving as support for these people establishing a relationship equality with them.

The goal of this Final Degree Project is the design, development and implementation of a progressive web application (PWA), called “Cantarranas”. From there, the volunteers who carry out the routes can write, publish and consult reports of the routes themselves. Aditionally, allow the workers of the organization to exploit the data of the reports, generating graphics with them, as well as keeping track of the active volunteers to facilitate and optimize the organization. On the other hand, it will serve as a small social network for the organization, which connects the volunteers of the different routes, with a space where they can share content between them and communicate with the administrators. In addition, it will also have a calendar where they can check the events, cleaning shifts and closing days of the food place.

These functionalities have been defined in various meetings with the head of the homeless program of “Solidarios para el Desarrollo”, attending to the needs of the program and the requests of volunteers.

The application has been developed at the Frontend level with the JavaScript framework Vue.js, supported by Nuxt.js and the Vuetify component library. The Backend has been implemented with Firebase, using its Hosting service as well as Realtime Database, to build the database in real time.

The last stage of this project has consisted of real tests, in which the application has been used by more than 60 volunteers, aiming to be used by the whole organization by Septmbre 2019.

Keywords

Homeless, Solidarios para el Desarrollo, Cantarranas, Progressive Web Application, PWA, JavaScript, Vue, Nuxt, Vuetify, Firebase.

Agradecimientos

Primero agradecer a toda mi familia el apoyo recibido desde el comienzo del grado. A mis padres especialmente, por el apoyo moral y económico.

A mi hermano Jesús, por las clases de programación desde primero y los consejos recibidos durante toda la carrera.

A mi tutor Miguel Ángel Mora, por todo el tiempo dedicado, por atreverse a tutelarme el proyecto desde la propuesta, por el ánimo transmitido especialmente en los comienzos, por la infinidad de consejos y por la gran disponibilidad mostrada.

A Alejandro Sierra, por poner la primera piedra del proyecto, enseñándome programación en primero de carrera y poniéndome en contacto con Miguel, cuando este proyecto era sólo una idea.

A todos los profesores del grado, incluidos los de la Universidad de Pisa donde hice el Erasmus. Especialmente gracias a todos los que me han aportado conocimientos básicos para realizar este proyecto con las asignaturas de programación, redes, sistemas informáticos y sistemas distribuidos.

A Marián, Pablo y Eduardo, los culpables de que hace 6 años conociera el increíble voluntariado que me ha llevado a realizar este proyecto.

A todos los voluntarios de Solidarios para el Desarrollo y al responsable del programa de Personas sin Hogar, Jesús Sandín. Por la inmensa labor que realizan y por su gran acogimiento e implicación en el desarrollo de la aplicación.

A todos mis amigos y compañeros de la universidad, llegar hasta aquí ha sido un auténtico trabajo en equipo.

Por último, gracias a la sociedad española y europea que, a través de sus impuestos, han financiado la gran parte de mis estudios en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid, ofreciéndome la oportunidad de recibir una educación universitaria pública de calidad. Siempre tendré una deuda con ellos y este proyecto es el primer paso para saldarla.

Gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

Según el último recuento de Personas Sin Hogar en Madrid, realizado en Diciembre de 2018 y organizado por el Ayuntamiento de Madrid, 2.772 personas se encuentran sin hogar en nuestra ciudad, 3.006 si contamos los Centros de Acogida de Inmigrantes, de las cuales 650 se encuentran en situación de calle [1]. Es decir, 650 personas duermen cada noche en las calles de Madrid.

Solidarios para el Desarrollo es una organización de ámbito nacional, fundada en el seno de la UCM por el profesor Jose Carlos García Fajardo, hace poco más de 30 años [2]. Entre sus programas de acción social, se encuentra el de Personas Sin Hogar, formado por grupos de voluntarios que cada noche, de lunes a jueves, recorren las calles de Madrid donde habitualmente se encuentran las personas sin hogar y, compartiendo café o caldo, tratan de vencer la soledad y establecer una relación de "igual a igual" con la persona, que potencie su autoestima y sirva de "puente" con la sociedad en favor de su integración en la misma.

Figura 1: Logo de Solidarios

Este TFG, ha sido propuesto por el alumno Carlos Iniesta, autor del mismo y voluntario en la organización desde hace 6 años. Con el fin de utilizar las últimas tecnologías para desarrollar una aplicación que potencie la acción social que se realiza en el programa de Personas Sin Hogar. Para ello, por un lado se han tratado de definir los requisitos funcionales y no funcionales de la aplicación y, en base a los mismos, escoger las tecnologías más adecuadas, realizando un análisis previo de las alternativas más importantes.

Si el tribunal lo desea, puede acceder a la aplicación desde https://www.cantarranas.es, y añadirla a su dispositivo para probarla. Podrá identificarse a través de una cuenta de google y después señalar que no es voluntario, lo que le dará acceso como invitado a las funciones más básicas de la aplicación. También, si lo desea, podrá solicitar por correo electrónico al alumno (carlos.iniesta@estudiante.uam.es), que se le concedan permisos de voluntario de manera temporal hasta el día de la defensa, con el fin de poder visualizar la aplicación.
Destacar que además de esta memoria, se han realizado dos documentos como manuales para los usuarios y administradores de la aplicación. Estos documentos no han sido incluidos en la memoria por su extensión, pero es posible acceder a ellos:

- Manual de Usuario
- Manual de Administrador

Por último, comentar también que el código desarrollado se encuentra en un repositorio privado de Github, al que los miembros del tribunal podrán solicitar acceso si lo consideran oportuno a través del correo electrónico del alumno.

1.2 Objetivos

El objetivo general del proyecto, podría describirse a grandes rasgos como la digitalización del programa de Personas Sin Hogar de Solidarios para el Desarrollo en Madrid, a través de una aplicación web que hace de CRM, como herramienta organizativa y de explotación de datos, y a su vez de red social interna, siendo un espacio de encuentro entre los voluntarios que participan en el programa.

Durante las reuniones concertadas con el director del programa, se captaron y definieron los requisitos que se exponen con más detalle en el apartado "3.1 Análisis y captación de requisitos" del presente documento. La primera necesidad que se trasladó por parte del director, fue sustituir el sistema existente para realizar los informes de ruta. Estos informes se escriben a través de grupos de Gmail, a los que el coordinador de cada ruta envía un correo resumiendo la ruta e informando del número de voluntarios, personas visitadas y personas dormidas, además de comunicar incidencias. Al realizarse de esta manera, cada informe se redactaba con un estilo distinto, y muchas veces se enviaba varios días después de realizar la ruta, dificultando la extracción y uso de los datos que incluía el informe. El primer objetivo de la aplicación consistía en sustituir este sistema por una herramienta ágil y útil, que permitiera redactar los informes mediante un formulario, facilitando así la explotación de los datos y simplificando su escritura.

Otro objetivo importante era conectar a los voluntarios de las diferentes rutas y días. Para ello se propusieron dos líneas, por un lado desarrollar un espacio donde pudieran compartir artículos, reflexiones, libros, proponer quedadas, etc. Y por otro lado, generar un repositorio con los informes de ruta publicados, accesible a todos los voluntarios, pudiendo leer los informes de otras rutas.

Por último, el tercer objetivo que se propuso abordar, consistía en la modernización y mejora de la organización del programa. En primer lugar, facilitando la comunicación entre los trabajadores y los voluntarios, especialmente con los coordinadores de las rutas. En segundo lugar, mediante un calendario interactivo que reflejara los turnos de limpieza del local y eventos que se organizan. Y en tercer lugar, proporcionando una herramienta de gestión de los voluntarios en activo a través de sus cuentas de usuario en la aplicación.

Alcanzar estos objetivos funcionales conlleva que la aplicación cumpla una serie de características técnicas, comenzando con la necesidad de ser multiplataforma, para llegar a todos los voluntarios sin importar el sistema operativo de sus dispositivos móviles, y permitiendo a su vez que sea utilizada por los trabajadores del programa desde un ordenador. Además, la aplicación debe ser intuitiva, para que pueda ser utilizada por los distintos perfiles de voluntarios sin suponer un esfuerzo extra. También debe tener un mantenimiento sencillo, al tratarse de un proyecto a largo plazo. Y todo esto se debe cumplir minimizando los costes, ya que al tratarse de una ONG no se obtendrán beneficios económicos, por lo que la inversión deberá ser mínima.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del Arte**

  Se analizarán las distintas tecnologías existentes con las que se podría haber realizado el proyecto, incluyendo las que finalmente se han utilizado. Comenzará con el estado de las aplicaciones multiplataforma y las alternativas con las que poder desarrollarlas. A continuación, se explorarán las opciones existentes para la realización del Frontend y del Backend.

  Por último, se expondrán algunos casos de éxito de aplicaciones que han sido desarrolladas para una ONG o con un fin social.

- **Diseño**

  Primero se enumerarán los requisitos funcionales y no funcionales definidos para la aplicación. En segundo lugar, se detallará el diseño planteado para la aplicación y se justificarán las tecnologías utilizadas para su desarrollo entre las alternativas expuestas en el Estado del Arte, en base a los requisitos expuestos.

- **Desarrollo**

  Se explicarán las pautas seguidas a lo largo de la construcción de la aplicación y las peculiaridades técnicas que contiene la misma. Se comentarán los apartados más importantes dentro de la aplicación y cómo se han llevado a cabo.

- **Pruebas y resultados**

  Se expondrán los resultados de las pruebas realizadas. Pasando por los planes de prueba que se realizaban en paralelo al desarrollo, para después destacar de manera singular las pruebas realizadas con usuarios reales.

- **Conclusiones y trabajo a futuro**

  Se desgranarán las conclusiones obtenidas de la realización del proyecto en todas sus fases. Además, se enumerarán las mejoras y nuevas funcionalidades que se plantean desarrollar sobre la aplicación, algunas de ellas propuestas por los propios usuarios.

- **Anexos**

  Se aportan además diversos anexos que complementan el presente documento. En ellos se explica cómo instalar la aplicación, se detallan las reglas de seguridad implementadas y se aporta un histórico de las fases y reuniones que se han producido con el director del programa de personas sin hogar de Solidarios.

2 Estado del arte

2.1 Aplicaciones multiplataforma

Conseguir romper las barreras entre plataformas, desarrollando una única aplicación compatible para cualquier dispositivo, sin importar su sistema operativo, es aún a día de hoy un auténtico reto tecnológico. Existen diversas alternativas para conseguir este objetivo y evitar desarrollar una aplicación diferente para cada plataforma, ahorrando tiempo y recursos en su desarrollo, además de facilitar su mantenimiento a futuro al tratarse de un único código a mantener.

Podemos clasificar las diferentes alternativas en cuatro grandes bloques que iremos desgranando, comentando las opciones que existen dentro de cada bloque, sus ventajas e inconvenientes y citando ejemplos de aplicaciones conocidas basadas en la tecnología en cuestión. [3]

2.1.1 Aplicaciones Híbridas

Las aplicaciones hibridas, están desarrolladas en general con HTML5, CSS y JavaScript al igual que una aplicación web común, pero son desplegadas a través de un componente nativo, un contenedor "Web View". Esto permite que se puedan distribuir de igual manera a las nativas, a través de las tiendas oficiales de cada plataforma a pesar de que realmente son aplicaciones web.
La gran diferencia se notará en el rendimiento y la experiencia de usuario respecto a las nativas, así como la posibilidad de acceder a las capacidades del dispositivo, aunque cada vez existen más librerías para poder utilizar el Bluetooth, GPS, cámara, etc.
Sin duda, la opción más conocida y utilizada para el desarrollo de aplicaciones hibridas es **Apache Cordova**, un marco de desarrollo móvil de código abierto, antes conocido bajo el nombre de phoneGap, hasta que en 2012 Adobe compró el proyecto y creó Cordova, dejando phoneGap como una versión independiente que trata de mantener alineada.
El funcionamiento de Cordova puede verse resumido en la Figura 2, se observa que su motor es una aplicación web, cargada en el dispositivo mediante un WebView y apoyada en los plugins de Cordova para poder utilizar los componentes del dispositivo móvil. [4]

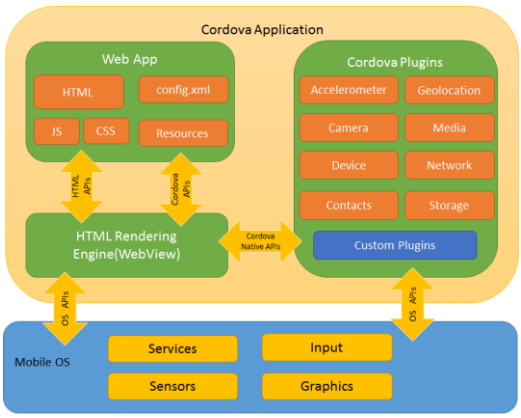


Figura 2: Arquitectura Cordova [4]

Es inevitable, cuando hablamos de ejemplos de aplicaciones híbridas que utilicen Apache Cordova, hablar de **Ionic**, uno de los frameworks mejor posicionados y más utilizados en

la actualidad a la hora de desarrollar aplicaciones de este tipo. Ionic utiliza el framework de JavaScript Angular para la "Web App" y Cordova para poder acceder a las funciones nativas de los dispositivos.
Grandes empresas como Diesel, Mclaren o McDonald's han optado por Ionic para el desarrollo de sus aplicaciones y los resultados son realmente buenos. Podemos destacar por ejemplo la aplicación MarketWatch, con más de un millón de descargas entre IOS y Android, que informa de noticias financieras en tiempo real, datos de los mercados y análisis de inversión. [5]

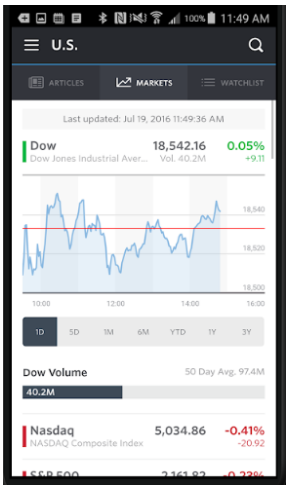


Figura 3: MarketWatch [5]

2.1.2 Aplicaciones Interpretadas

Estas aplicaciones tienen la característica de ser implementadas en un lenguaje base, del cual una parte se traduce a nativo y otra es interpretada en tiempo de ejecución. Esto permite que la experiencia de usuario sea muy similar a las nativas, y se puedan hacer uso de las características hardware del dispositivo de forma nativa. Cierto es que para ello, no es posible utilizar exactamente el mismo código para todas las plataformas, sino que deben seleccionarse los componentes específicos para cada una.
En esta dirección aparecen múltiples frameworks entre los que destacan cuatro:

- **Appcelerator Titanium**: el más antiguo de los cuatro, presume de reutilizar entre el 60%-90% de su código entre plataformas, de contar con más de 660.000 desarrolladores y 75.000 aplicaciones implementadas. Se desarrolla con Javascript y se apoya en Tituanium Studio como entorno. Además ofrece también la posibilidad de usar Alloy, un marco sobre Titanium que facilita implementar la arquitectura MVC (Modelo Vista Controlador). [6]

- **NativeScript**: lanzado en 2015, no tardó en ganar popularidad. Utiliza Javascript, CSS y XML pero destaca por soportar también TypeScript, Angular y desde hace poco Vue.js. Lo que lo convierte en especialmente atractivo para desarrolladores web que dominen estos frameworks. Consigue mejores tasas de reutilización de código que Titanium, superando el 90%, siendo necesarios sólo algunos retoques para que el desarrollo funcione de manera prácticamente nativa en Andoid e IOS. Grandes empresas han apostado por NativeScript como Deloitte, Verizon, SAP o la empresa de textil Puma, que tiene sus aplicaciones de e-commerce desarrolladas con este framework. Para entender su funcionamiento, nos aporta en su página oficial la siguiente figura: [7]

Figura 4: NativeScript [7]

- **React Native:** apareciendo el mismo año que NativeScript (2015), es la que más se ha consolidado en el mercado, seguramente por tener detrás a Facebook y utilizar su framework ReactJS, que funciona con Node.js y NPM. React Native no consigue reutilizar tanto código como NativeScript (<90%), pero debido a la gran comunidad que le respalda, resulta sencillo encontrar los componentes más adecuados para Android e IOS. Y es que React Native es en esencia un conjunto de componentes de ReactJS que tienen su equivalente nativo, su funcionamiento se puede resumir con los tres módulos de la Figura 5. La lista de aplicaciones más importantes desarrolladas con esta tecnología no sólo se compone por Facebook e Instagram, a ella se suman Airbnb, Uber Eats, Pinterest e incluso la aplicación de Skype para Android e IOS. [8]

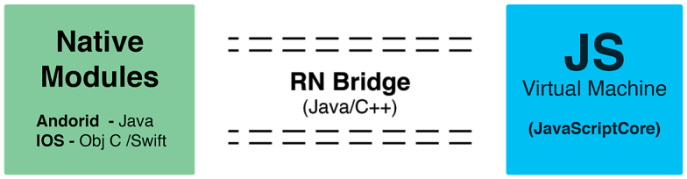


Figura 5: React Native [9]

- **Flutter:** El más moderno de los analizados, su primera versión estable, Flutter 1.0, fue presentada hace unos meses, en diciembre de 2018 y, el 7 de mayo de 2019 se publicó Flutter 1.5. Este Framework creado por Google tiene un gran potencial. Se trabaja con un lenguaje desarrollado por la propia Google llamado Dart, orientado a objetos y con una curva de aprendizaje sencilla para los que hayan trabajado con C#, C++ o Java. A pesar de su juventud, Google ya tiene aplicaciones desarrolladas con Flutter como Google Ads, pero sobretodo destaca que, Alibaba, la empresa de comercio online más grande del mundo, tiene su aplicación para Android e IOS con esta tecnología y posee más de 50 millones de descargas.[10]

Figura 6: Appcelerator, Native Script, ReactNative and Flutter

2.1.3 Aplicaciones Generadas

Llamadas así ya que se desarrollan con las herramientas y lenguajes específicos de cada framework, para luego compilarse de manera nativa en cada plataforma. Esto provoca que la curva de aprendizaje sea algo pronunciada, pero permite una reutilización de código prácticamente completa (>90%), pudiendo acceder a todas las características del dispositivo. En este sentido podemos destacar algunas tecnologías como **RubyMotion**, en la que se desarrolla con el framework Ruby, **Delphi 10 Seattle** de Embarcadero, en la que se utiliza un entorno visual propio para desarrollar aplicaciones y, seguramente la más famosa de este género, **Xamarin**, en la que nos vamos a centrar un poco más:

- **Xamarin:** esta tecnología fue adquirida por Microsoft en 2016 que facilitó su uso básico de manera gratuita. En Xamarin se desarrolla con C#, y cuenta con un IDE propio llamado Xamarin Studio, aunque actualmente puede integrarse mediante herramientas específicas con Visual Studio también de Microsoft. Dentro de Xamarin podemos diferenciar dos ramas, una conocida como 'Xamarin Classic' donde la parte de lógica es compartida entre aplicaciones, pero la interfaz de usuario se desarrolla por separado con Xamarin.Android, Xamarin.Ios y/o Xamarin.Mac, reduciendo las tasas de reutilización de código pero consiguiendo una experiencia más nativa. La otra rama es Xamarin.Forms, donde se utiliza el mismo código, casi al 100% para las distintas plataformas, renunciando eso sí, a que la aplicación parezca visualmente nativa. Microsoft ha desarrollado con Xamarin su aplicación móvil para Azure, pero también otras empresas como UPS, empresa Americana de transporte de paquetes, o Fox Sports, han elegido esta tecnología para sus aplicaciones.[11]

Figura 7: Xamarin

2.1.4 Aplicaciones Web

Aunque ya las hemos mencionado, al tratarse del origen de las aplicaciones hibridas, merecen un apartado específico debido a su gran uso de manera individual. Tienen como principal característica que se ejecutan desde un navegador. Esto supone una gran ventaja, puesto que cualquier dispositivo con un navegador y conexión a internet podrá ejecutarlas pero, al mismo tiempo, supone la desventaja de que su rendimiento se verá afectado por la interacción-cliente servidor, y la experiencia de usuario quedará lejos de la nativa.
Estas aplicaciones están desarrolladas con HTML, CSS y JavaScript, y aunque su uso es de hace tiempo, en los últimos años han ido cogiendo fuerza, mejorando las experiencias de usuario y la comunicación con los componentes hardware de los dispositivos, llegando a convertirse en aplicaciones de gran usabilidad. En este marco se sitúa el concepto de Progressive Web Applicactions, conocido por sus siglas en inglés PWA que vamos analizar:

- **Progressive Web Applications:** este concepto no es tan novedoso, en 2015 Alex Rusell enumeraba los requisitos que debía cumplir una PWA. Estos, en líneas generales, consistían en aprovechar las nuevas tecnologías de los navegadores para que experiencia de usuario sea inmersiva como en las nativas, no necesite necesariamente de conexión a internet para mostrar contenido a haciendo uso de la caché, tenga una respuesta rápida a las interacciones y

además pueda hacer uso de las características de los dispositivos como él envío de notificaciones push o utilizar el GPS para obtener la ubicación del dispositivo. Además por supuesto debe ser instalable, cosa que conseguimos añadiendo a la pantalla de inicio del dispositivo el acceso directo a la web. [12]
Actualmente, los grandes navegadores (Chrome, Safari, Edge, etc.) están avanzando en aras de dar soporte a todos los requisitos de estas aplicaciones, que cuentan ya con grandes ejemplos, como las redes sociales Instagram, Twitter y Flipboard, que poseen versiones PWA prácticamente idénticas a sus versiones nativas.

Figura 8: Twitter PWA

2.2 Frontend Web

Analizadas las distintas alternativas existentes para el desarrollo de aplicaciones multiplataforma, podemos observar que la mayoría de ellas tienen en común el uso de los lenguajes web por excelencia: HTML, CSS y JavaScript.
El ya más que conocido lenguaje básico de la World Wibe Web, **HTML (HyperText Markup Language),** se encuentra actualmente en su versión HTML5 lanzada en 2014, con nuevas etiquetas y atributos, además de nuevas APIs para la validación de formularios, eventos sin conexión, utilización de la cámara, etc. En general más preparado para los retos actuales de la Web. Además cabe destacar que también existe la versión XHTML5, basada en XML.
Por otro lado, **CSS (Cascading Style Sheets)**, también es de sobra conocido por ser el lenguaje utilizado para dotar de diseño visual y presentación a los documentos HTML o XHTML. Su versión más reciente es CSS3.1, pero gracias a su división en módulos, se van desarrollando e incorporando mejoras.
JavaScript por su parte es el lenguaje encargado de la lógica y acciones de la web. Es un lenguaje interpretado por los navegadores, débilmente tipado y orientado a objetos, se trata de una implantación de ECMAScript, que se encuentra en su décima versión, ES10.
El caso de Javascript requiere un especial análisis, ya que para el desarrollo Frontend se acostumbra a utilizar un frameworks sobre el lenguaje, y su elección es una decisión muy importante para el desarrollador. La lista de frameworks de Javascript para Frontend es cada vez más extensa, destacan Ember, Polymer, BackBone, Preact, React, Vue.js y Angular, centraremos el análisis en los tres últimos ya que son los más utilizados.

Para realizar las comparaciones, se ha utilizado como apoyo fundamental los resultados de "The State Of Javascript" en su edición de 2018, una encuesta que recoge el uso, interés y satisfacción de más de 20.000 desarrolladores de la comunidad JavaScript. [13]

- **React**
  Ya mencionado cuando hablamos de React Native dentro las aplicaciones interpretadas (2.1.2), este framework de código abierto, mantenido por la comunidad y por Facebook, es el más utilizado actualmente y está en constante crecimiento. De los encuestados, los han utilizado un utilizado un 71.5%, de los cuales un 90% afirman que lo volverán a usar.
  La comunidad de JavaScript destaca de React la elegancia de su paradigma y patrones de desarrollo, así como su riqueza en paquetes y documentación, en cambio se le achaca tener una curva de aprendizaje compleja y estar sobrecargado.

- **Vue.js**
  Algo menos conocido, aunque dentro de la comunidad dejó de serlo hace tiempo. De hecho, sólo un 1% de los encuestados dicen no haber oído hablar de él, pero lo más interesante es que un 46.6% afirman estar interesados en aprenderlo, convirtiéndolo en el lenguaje que más desarrolladores quieren aprender, sumado a que es de código abierto, asegura un crecimiento futuro prometedor. Además es el segundo que más desarrolladores volverían a utilizar tras hacerlo anteriormente, de hecho sólo un 2.8% de los encuestados no pretenden volver a utilizarlo.
  Lo que más se destaca de Vue es que su curva de aprendizaje es fácil, posee una buena documentación, además de la simpleza y ligereza del framework. Sin embargo, se critica que tenga un estilo de programación torpe y que se encuentra algo limitado por falta de funcionalidades.

- **Angular**
  También se mencionó anteriormente al hablar de Ionic, en el análisis de las aplicaciones híbridas (2.1.1) y, dentro de NativeScript en las interpretadas (2.1.2). Este framework desarrollado en TypeScript, de código abierto y mantenido por Google, es conocido cómo Angular 2, puesto que es el predecesor de AngularJS. Lo que más sorprende de sus resultados en la encuesta, es que un 58.6% de los desarrolladores que lo han utilizado dice que no lo hará de nuevo y, un 31.8% ha oído hablar de él pero no está interesado en aprenderlo. Esto demuestra la impopularidad que ha ido cosechando Angular en la comunidad de JavaScript, aunque hay que destacar que muchas empresas siguen apostando por él, seguramente gracias a tener a un gigante como Google detrás.
  Los desarrolladores destacan que Angular es un framework poderoso y muy completo en funcionalidades, además al igual que los dos anteriores posee una buena documentación y un paradigma de programación adecuado. Por el contrario, se le critica la sobrecarga, complejidad y curva de aprendizaje.

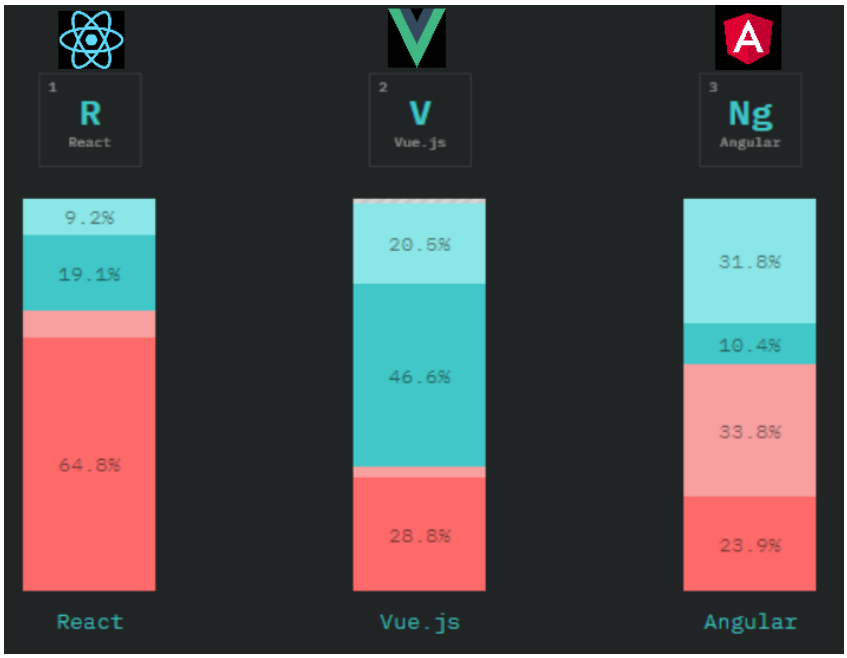


Figura 9: "The State Of Javascript" (React, Vue, Angular) [13]

Cabe destacar, que los tres framework expuestos permiten el desarrollo de aplicaciones **SPA (single-page application)**. Estas tienen la peculiaridad de que al ser ejecutadas, el servidor web entrega al navegador el cliente de Javascript con todo el contenido HTML, CSS y Javascript y los recursos para ejecurarlo de una vez. Así no se hace necesario cargar de nuevo una pagina cuando el usuario lo requiere, simplemente se reemplazan unos componentes por otros mediante llamadas que el cliente realiza a la API. En la Figura 10 podemos ver el funcionamiento, que supone un salto de calida para las aplicaciones web y el desarrollo frontend, puesto que se consigue agilizar la navegacion mejorando la experiencia de usuario. [14]

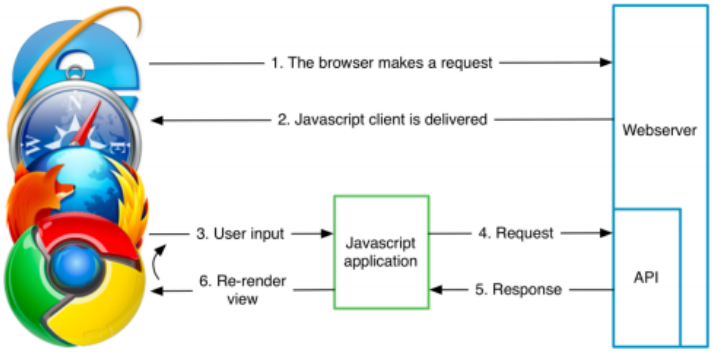


Figura 10: Single-Page Application [14]

2.3 Backend Web

Una vez expuesto el estado de las distintas alternativas para realizar el Frontend Web, se hace indispensable hacer lo propio con las opciones existentes para realizar el Backend. Es fundamental elegir la opción correcta para el desarrollo de una aplicación, ya que es el encargado de tratar los datos, protegerlos y comunicarse del lado del servidor. Existen múltiples tecnologías para la realización de Backends que se ha decidido clasificar en tres grandes opciones:

- **Backend "Clásico":** es el programado típicamente con PHP y SQL, o también JAVA o Python entre otros. Permite la posibilidad de desarrollar un backend propio y personalizado. Al ser lenguajes que ya tienen un gran recorrido, PHP por ejemplo fue lanzado en 1995, poseen bibliotecas ricas de instrucciones y documentación, son más simples y están soportados por la mayoría de servicios de alojamiento de internet. En su contra, consumen muchos recursos del lado del servidor, ya que cada conexión genera un hilo nuevo que ocupa una cierta memoria, lo que ante conexiones concurrentes ralentiza la experiencia de usuario. Esta opción de Backend es la que tienen muchas grandes empresas en sus sistemas informáticos o aplicaciones, Wikipedia y Yahoo Inc por ejemplo, utilizan PHP.
  Dentro de esta modalidad, podríamos incluir también los backend desarrollados con ASP.NET, la plataforma de desarrollo web respaldada por Microsoft y utilizada por numerosas empresas.

- **Backend con Node.js:** es complicado definir realmente que es realmente Node.js, en su página web se define a sí mismo como "un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome". [15] Comenzó su andadura en 2009 y es un auténtico ecosistema que permite el uso de JavaScript en el lado del servidor, a través de sus librerías, su escalabilidad, su arquitectura basada en eventos y funciones asíncronas, además de su conocido gestor de paquetes NPM (Node Package Manager). Su gran ventaja es la capacidad de soportar miles de conexiones concurrentes gracias a funcionar mediante eventos. El uso de Node.js está avalado por grandes empresas como Netflix, Paypal, Linkedin y NASA.
  Por supuesto, al igual que en Frontend, existen distintos frameworks de JavaScript para el Backend con Node.js. Algunos de ellos son Meteor, Socket.io, Koa, Sails, Next, aunque el más utilizado en la actualidad es Express. Estos frameworks también son analizados en la encuesta global "The State of Javascript" [13], donde arrasa Express en cuanto a uso y satisfacción, destacando su estabilidad, curva de aprendizaje sencilla, riqueza de paquetes, ligereza y buena documentación. Y valorando negativamente su estilo de programación torpe.

- **BaaS (Backend as a service):** se trata de un servicio que proporciona a los desarrolladores el backend en sí, pudiendo conectar las aplicaciones a la nube, alojar las bases de datos, gestionar los usuarios, integración con proveedores de identidad, servicios de análisis o notificaciones push. A diferencia de los otros dos modelos, no requiere desarrollar mediante código el backend si no que se "contrata" y se configura para la aplicación, aunque una solución muy común es combinar una parte de backend desarrollado y otra parte mediante BaaS. Se conoce también como MBaaS, puesto que se utiliza principalmente en aplicaciones móviles, tanto web como nativas. Uno de los pioneros fue Parse, que se asentó con fuerza en el mercado, fue adquirido por Facebook en 2013. Finalmente la propia Facebook decidió cerrarlo en 2017, dejando su contenido como código libre, fenómeno que provoco que surgieran distintas alternativas para que los desarrolladores pudieran migrar sus servicios de Parse, como son Kumulos, Sashido, Mongodb o Black4app entre otros, que ofrecen aun planes gratuitos para migrar o contratar sus servicios como backend. [16]

Además existen otras alternativas como Kinvey, otro de los pioneros del mundo Baas, o Azure Services, creado por Microsoft también con la tecnología de Parse. Pero seguramente los dos más completos e implantados y que han dado un paso más en el mercado, son Firebase de Google y AWS de Amazon (destacando AWS Lamda y AWS Amplify). Estos Mobile Backend as a Service van un paso más ofreciendo prácticamente todas las funcionalidades de un backend, desde la seguridad y alojamiento de la base de datos a la autenticación, las notificaciones push, la persistencia de datos, análisis y gestión de errores, etc. Todo ello mediante micro servicios a los que podemos acceder de manera limitada con una suscripción gratuita, suficiente para aplicaciones pequeñas. (Firebase Pricing [17], AWS Pricing [18])

2.4 Aplicaciones móviles en el ámbito de las ONG

En los últimos años, han surgido infinidad de proyectos que, apoyados en el desarrollo de una aplicación móvil, tienen un fin social. Y es que la tecnología es sin duda una de las mejores armas para generar impacto e impulsar acciones sociales que cambien el mundo logrando una sociedad más justa. Son numerosos los casos de éxitos desarrollados con distintos tipos de tecnología, alguno de ellos son:

- **ShareTheMeal**: Impulsada por la ONU, consigue "compartir" tu comida con quienes más lo necesitan, pudiendo hacer donaciones desde 40 céntimos desde la app eligiendo el proyecto al que va destinando. Ya han conseguido compartir más de 40 millones de comidas. Esta aplicación está desarrollada de manera nativa tanto para Android como para IOS.
- **Hacesfalta.org:** Creada por la Fundación Hazloposible, conecta voluntarios interesados y ONG. Ambas partes deben registrarse en la aplicación, las ONG publican sus proyectos y los voluntarios pueden buscar entre ellos filtrando por su interés de manera sencilla. La aplicación está desarrollada de manera hibrida y cuenta con más de diez mil descargas.
- **CharityBuzz y Charity Miles:** ambas son ejemplos de éxito de aplicaciones de con un fin social, la primera es una plataforma de subastas con fines solidarios, en la segunda puedes donar al proyecto que elijas el dinero que consigues a través de hacer kilómetros, corriendo o en bicicleta. Además, Charity Miles cuenta ya con más de medio millón de descargas y ambas están desarrolladas de forma nativa.
- **1Heart1Tree:** esta aplicación está enfocada a la reforestación del planeta. El proyecto fue creado por la artista belga afincada en Francia Naziha Mestaoui, Permite mediante donaciones replantar árboles en proyectos concretos que han sido seleccionados por todo el mundo. Han conseguido replantar ya 100.000 árboles. La aplicación es puramente web, podemos acceder a ella a través de https://www.1heart1tree.org/ desde cualquier dispositivo, y si lo deseamos, añadirla a nuestra pantalla de inicio para tener una experiencia de aplicación más nativa.
- **Voltimers:** por último, podemos destacar el caso de esta Start-Up de origen catalán. Que bajo el lema "horas que salva vidas", permite realizar aportaciones a proyectos solidarios haciendo lo que más te gusta o mejor se te da. Permite que los usuarios propongan proyectos o actividades a un módico precio, cuya recaudación se destina a fines solidarios. Su aplicación es hibrida, y está desarrollada con Cordova.

3 Diseño

Una vez analizado el estado actual de las alternativas que existen para desarrollar aplicaciones multiplataforma y, mencionados algunos ejemplos del impacto de las mismas en proyectos sociales. Se hace necesario enumerar los requisitos definidos para la aplicación, y en base a ellos, justificar la elección de las tecnologías utilizadas en el desarrollo de la aplicación y el diseño propuesto para la misma.

3.1 Análisis y captación de requisitos

3.1.1 Requisitos funcionales

Los objetivos funcionales de la aplicación, definidos conjuntamente con el director del programa de personas sin hogar de Solidarios, podrían resumirse en los siguientes puntos:

1. Desarrollar una herramienta ágil para redactar los informes de ruta que realizan los voluntarios cuando hacen el recorrido visitando a las personas sin hogar.
2. Generar un repositorio de estos informes, accesible por todos los voluntarios y con posibilidad de realizar consultas filtradas de manera sencilla.
3. Posibilidad de obtener y tratar los datos concretos de los informes de ruta, como el número de voluntarios o personas visitadas, con el fin de sacarles partido mediante la generación de gráficos para poder conocer el estado y alcance del programa de una manera eficiente.
4. Generar un espacio de encuentro entre todos los voluntarios de las distintas rutas, donde puedan compartir opiniones y propuestas.
5. Dotar también de un canal de comunicación directa entre los coordinadores de las rutas y los trabajadores del programa, para la gestión de incidencias y seguimientos de las personas sin hogar.
6. Desarrollar una sección con un calendario donde publicar los turnos de limpieza y eventos programados de la organización.
7. Dividir las funcionalidades de la aplicación por roles asociados al usuario, distinguiendo entre: Invitados, Voluntarios, Coordinadores y Administradores.
8. Obtener un registro de los voluntarios del programa que sea accesible y gestionable, permitiendo editar y eliminar usuarios.

3.1.2 Requisitos no funcionales

Con la misión de cumplir los objetivos funcionales mencionados, se definieron una serie de requisitos técnicos básicos que debía cumplir la aplicación y que serían claves a la hora de decidir qué tecnologías utilizar para su desarrollo.

1. **Multiplataforma**. Debido a que entre los voluntarios existen todo tipo de dispositivos: móviles Android e IOS, algunos más acostumbrados al uso de tabletas y, los trabajadores del programa requerían utilizar la aplicación desde un ordenador de sobremesa con el que trabajan desde la oficina.
2. **Ligera**. Ya que muchos de los voluntarios nos comentaron que apenas tienen espacio para instalar nuevas aplicaciones, un requisito básico era minimizar el espacio de memoria que pudiera ocupar la aplicación, permitiendo su instalación y uso en móviles con un espacio libre de entorno a 50MB.
3. **Segura**. Intentar evitar el manejo de contraseñas, explorando la posibilidad de utilizar un proveedor de identidades externo como Google o Facebook para la

autenticación. Además de que los datos de los resúmenes sólo pudieran ser consultados por los voluntarios y demás miembros del programa.

4. **Intuitiva**. Tratar de que sea los más accesible posible debido a los diferentes perfiles de usuarios, algunos no muy acostumbrados a utilizar aplicaciones. No sobrecargarla de funcionalidad al ser la primera versión.
5. **Fácil mantenimiento**. No utilizar tecnologías experimentales que requieran de un mantenimiento costoso o con un futuro incierto. Escoger tecnologías nuevas y con proyección pero con cierto peso de cara a un mantenimiento a medio y largo plazo.
6. **Mínimos costes**. Sobretodo hasta no tener segura su implantación y beneficios, tratar de minimizar los costes al tratarse de una ONG, especialmente en el hosting.

Estos requisitos, fueron factores fundamentales a la hora de decidir las tecnologías que conformarían el proyecto y definir su diseño. La necesidad de que fuera multiplataforma, fue el primero en marcar el camino, alejando el proyecto de las aplicaciones nativas, teniendo en cuenta que al ser un único desarrollador, y que se busca un mantenimiento sencillo, era complicado construir varias aplicaciones paralelamente, que fueran de escritorio Windows, Linux o Mac OS, y dispositivos móviles y tabletas Android e IOS.

3.2 Tecnologías utilizadas

Atendiendo al estado del arte y a los objetivos que se desean alcanzar con el desarrollo de la aplicación, es el momento de exponer el diseño y tecnologías que se han escogido para su desarrollo.

La aplicación finalmente se ha realizado como una aplicación web, desarrollando una PWA, que se ejecuta en modo SPA, mediante el framework de JavaScript Vue.js, apoyándose en Nuxt.js y la librería de componentes Vuetify. Además, para el Backend se optó por el uso de Firebase, escogiendo su funcionalidad Realtime Database para la base de datos y sus reglas de seguridad para limitar el acceso a la misma. El hosting también se realiza con Firebase, pero para personalizar el dominio se adquirió uno con 1&1 IONOS: "*cantarranas.es*".

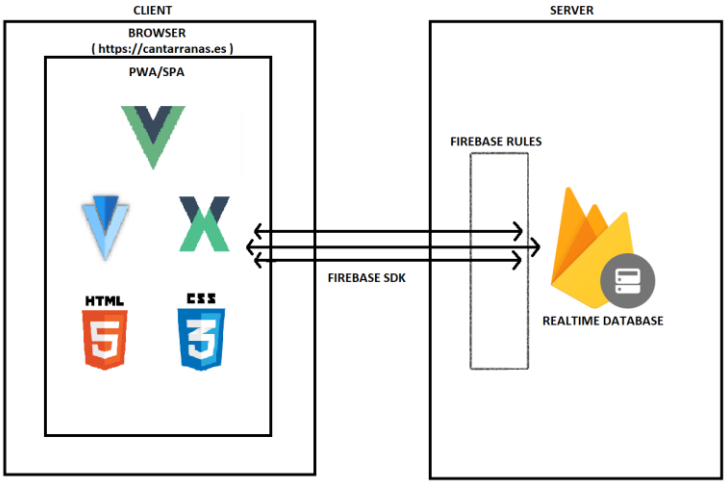


Figura 11: Arquitectura básica de la aplicación

3.2.1 PWA

La decisión de realizar finalmente una aplicación web, y en concreto una PWA, atendía principalmente a los requisitos de que fuera totalmente multiplataforma, ya que permite su ejecución con cualquier dispositivo que posea un navegador. También que fuera ligera, ya que al instalarse se trata de poco más que un acceso directo ocupando menos de 500KB.

Por otro lado, al tratarse de un único código para las diferentes plataformas, no sólo se facilita la posibilidad de realizarse por un único desarrollador, si no que hace más viable su mantenimiento posterior. Adicionalmente, se ajusta al objetivo de minimizar costes, puesto que no es necesario publicar la aplicación en las tiendas de las respectivas plataformas, lo que requeriría de una licencia de desarrollador que cuesta 25$ en el caso de Android y 99$ en el de IOS.
Otra ventaja destacable de la elección de una PWA, es que no requiere que el usuario actualice la aplicación para tener la última versión de la misma, basta con refrescarla. Con ello se consigue que todos los usuarios tengan la aplicación alineada y puedan realizarse mejoras de manera eficiente y efectiva. Por último, resaltar que al ser PWA, se consigue una mejor experiencia de usuario frente a las 'web app' tradicionales y se consumen menos datos, es posible cargar la aplicación aun estando sin conexión gracias a su uso de la caché y permite de una manera sencilla su "instalación", haciéndola tan accesible como cualquier otra aplicación.

3.2.2 Vue.js + Nuxt.js + Vuetify

Como se analizaba en el estado del arte, Vue.js es el segundo framework de JavaScript más usado por la comunidad de JavaScript y el que más desarrolladores tienen interés en aprender. Esto lo convierte en una apuesta segura a futuro donde irá creciendo, lo que asegura que una aplicación desarrollada con este framework tenga un mantenimiento viable con posibilidades de seguir mejorándola. Además, es el framework mejor valorado en cuanto a curva de aprendizaje, lo que resultaba importante para no perder demasiado tiempo en la etapa de formación y poder dedicarlo a construir una aplicación lo más completa, estable y funcional posible.
Es importante destacar que la elección de Vue.js no fue sola, se decidió utilizar en conjunto con Nuxt.js, un framework de alto nivel que facilita el desarrollo de aplicaciones Vue Universales, Generadas o de una sola página (SPA). Nuxt.js consigue que las configuraciones del lado del servidor sean más agradables con características como los middleware, layout, datos asíncronos, plantillas, etc. [19]. Los middlewares resultan de gran utilidad para cumplir con el requisito de limitar las funcionalidades de la aplicación mediante los roles. Nuxt.js trabaja con los componentes más importantes del ecosistema de Vue, como son:

- **Vue Router:** encargado de facilitar la navegación entre los componentes, enlazando las URL de los mismos e indicando a Vue donde mostrarlos. Es especialmente útil en SPA, permite mapear las rutas y pasar parámetros entre ellas, lo que ha resultado fundamental en el desarrollo de la aplicación.
- **Vuex:** se autodefine como una biblioteca de patrones de administración del estado. Inspirada en Flux, Redux y Elm Architecture pero creada específicamente para Vue.js, aprovechando su reactividad. Vuex simplifica la comunicación entre los componentes definiendo una arquitectura cuyo núcleo es el "store", donde se guarda el "state", que no se puede modificar directamente si no, a través de "actions", que se comunican con los servicios necesarios y se emiten desde la vista. La lógica para cambiar el estado se declara en las "mutations". La Figura 12 explica de una manera gráfica la arquitectura Vuex, que ha definido la arquitectura propia de la aplicación en cuanto a su manejo del estado. [20]
  Vuex se antojaba fundamental para para poder guardar el objeto de usuario en el "store" una vez se identificaba. Pudiendo ser accesible así desde cualquier componente o página.

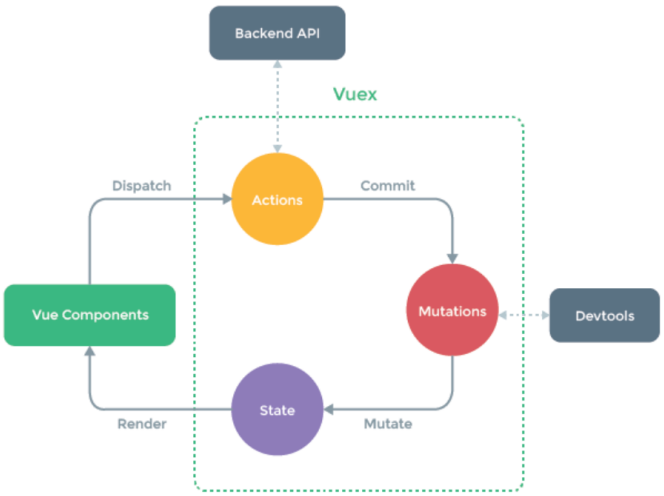


Figura 12: Vuex Architecture [20]

El funcionamiento interno de Nuxt está basado en el uso de Webpack con Vue-loader y Babel Loader. Webpack es el encargado de agrupar y empaquetar los recursos y archivos para su uso en el navegador. Vue-Loader el encargado de cargar los paquetes, permite crear los componentes de Vue en el formato SFC (Single-File Component), donde el HTML, el JavaScript y el CSS de un componente se escriben en el mismo archivo (Figura 16). Por último, Babel Loader es un paquete que permite transpilar los archivos JavaScript utilizando Babel como compilador. Gracias a este funcionamiento, se consigue organizar y minimizar el código, lo que facilita cumplir los requisitos de fácil mantenimiento y lograr que la aplicación sea lo más ligera posible.

Una vez decidido el uso de la combinación de Vue.js con Nuxt.js, llega el momento de elegir la librería de componentes con la que nutrir la aplicación. Es cierto que podemos combinar componentes de varias librerías, pero es recomendable utilizar principalmente una, así tu aplicación será más robusta y coherente tanto visualmente como internamente. Existen numeras fuentes de componentes y temas para Vue, las más importantes y completas son Quasar, Element, Boostrap-Vue, Buefy o Vue Material, aunque la más utilizada por la comunidad Vue es Vuetify. Este framework es el más completo en cuanto a componentes y temas, además tiene una gran soporte y se actualiza constantemente, con nuevos componentes y mejorando los existentes. Su documentación está repleta de ejemplos y cuenta con una gran cantidad de proyectos en los que se ha utilizado. Además posee soporte para SPA y PWA, y sigue al pie de la letra el estándar de Material Design [21]. Todas estas características, convierten a Vuetify en la mejor opción para que la aplicación cumpliera el objetivo de ser intuitiva y con un mantenimiento viable a futuro.

Figura 13: Vuetify

3.2.3 Plantilla y diseño inicial

Un factor determinante del aspecto final de la aplicación fue la plantilla inicial que se decidió utilizar, que incluye Vuetify y Nuxt.js. Esta plantilla definió la estructura de carpetas en las que organiza el código, la ideal para Nuxt, y el aspecto inicial de la aplicación. La plantilla se encuentra en github [22], y su instalación es realmente sencilla,

basta con descargarla y ejecutar los comandos que indica en su archivo “README.md”. Además permite el configurar que se ejecute en modo SPA, simplemente indicándoselo en el archivo “nuxt.config.js”.
Si ejecutamos dichos comandos, veremos el aspecto de la plantilla, que contiene básicamente un layout por defecto en tema oscuro, con un “Navigation Drawer” (menú lateral), con dos opciones, así como algunos botones en una “toolbar” (barra superior) con distintas funcionalidades. Además de un componente “<v-card>” (componente de Vuetify similar a una tarjeta) con información de la plantilla.

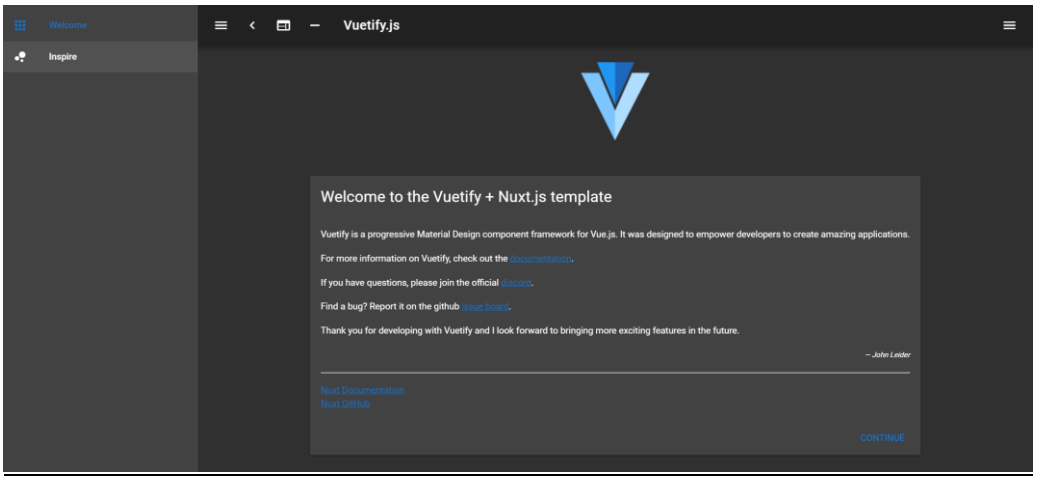


Figura 14: Plantilla Inicial

Esta plantilla, definió la estructura organizativa de los archivos, la ideal para Nuxt.js, que se ha mantenido durante el desarrollo del proyecto facilitando tener un código organizado y optimizado.

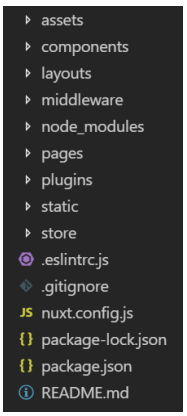


Figura 15: Estructura de archivos

Por otro lado, se decidió optar por seguir una arquitectura común durante el desarrollo de la aplicación. Las páginas, que se encuentran en “/pages”, simplemente son contenedores de los componentes necesarios, almacenados en “/components”, que son los que poseen la lógica. Además, utilizando el formato SFC, para los archivos “*.vue”.
Un ejemplo sencillo de esta arquitectura podría ser un componente llamado ‘**Hello.vue**’ que simplemente muestra el contenido de la variable “greeting”, junto con el texto “World!” dentro de un parrafo, con el texto alineado y el tamaño de la letra “2em”. Y una página ‘**hello.vue**’, que tan sólo importa el componente Hello.vue y lo situa en el “template”:

```vue
<template>
  <p>{{ greeting }} World!</p>
</template>

<script>
export default {
  data() {
    return {
      greeting: "Hello"
    };
  }
};
</script>

<style scoped>
p {
  font-size: 2em;
  text-align: center;
}
</style>
```

Figura 16: Componente 'Hello.vue'

```vue
<template>
  <Hello></Hello>
</template>

<script>
import Hello from "~/components/Hello.vue";

export default {
  components: {
    Hello
  }
};
</script>
```

Figura 17: Página 'hello.vue'

El resultado, sobre la plantilla inicial, sería el siguiente:

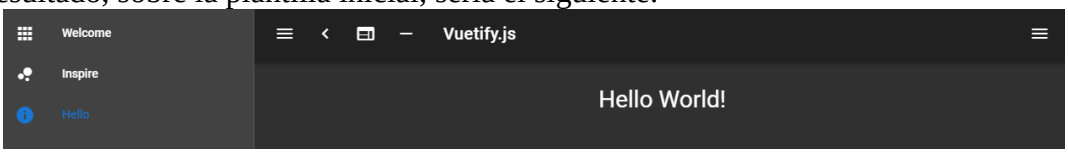


Figura 18: Resultado Hello World!

Al ser un ejemplo sencillo, no se aprecian directamente las ventajas de utilizar esta arquitectura o diseño, que a parte de conseguir tener un código organizado y de mayor legibilidad, permite la reutilización de componentes entre páginas.

3.2.4 Firebase

En cuanto al Backend, la elección fue usar un MBaaS, principalmente para poder dedicar más tiempo a la parte Frontend y desarrollar una aplicación lo más útil posible sin perder el tiempo en desarrollar un Backend propio, que además no llegaría a ser tan completo. Entre las opciones que se expusieron en el apartado 2.3 Backend Web, se optó por utilizar Firebase por múltiples razones: [23]

- Documentación y proyectos de referencia. Se valoró que la documentación de Firebase es muy completa, explica paso a paso como añadir Firebase a tu aplicación web, con ejemplos e incluso videos explicativos. Además, se encontraron numerosos proyectos donde se utilizaban Vue en conjunto con Firebase, sirviendo de referencia y, sumado al respaldo de Google, convertía la elección en una apuesta segura con vistas a futuro.
- Base de datos en tiempo real. Te permite alojar y manejar tu base de datos de una manera sencilla y en tiempo real, a través de Realtime Database. Se explica más en detalle en el apartado 3.3 Base y modelo de datos.
- Seguridad de los datos. Permite escribir reglas de seguridad para tu base de datos, que tienen un papel importante en el diseño final de la aplicación, puesto que todo

acceso a la base de datos, es evaluado mediante estas reglas que conceden o deniegan el permiso, asegurando parte del requisito de seguridad definido.

- Autenticación. Firebase Auth es el servicio de autenticación de Firebase, que permite utilizar otros proveedores de autenticación como Google, Facebook, Twitter, Microsoft, Yahoo o Github. También permite autenticación clásica mediante correo electrónico y contraseña. Esta característica facilitaba cubrir la otra parte del requisito de seguridad, evitando que la aplicación gestionara contraseñas.
- Analítica. Este servicio resultaba especialmente útil de cara a la evaluación de los resultados de las pruebas, ya que permite consultar el uso de tu aplicación de manera ágil.
- Notificaciones Push. Mediante su servicio Firebase Cloud Messaging, te da la posibilidad de implementar notificaciones push a los usuarios, lo cual te da la posibilidad de que la aplicación sea más completa y útil.
- Hosting. Permite de manera gratuita alojar tu aplicación en los servidores de Google y conectarla a un dominio personalizado, al que aprovisiona y configura un certificado SSL. Además el contenido se almacena en caché en los SSD de los servidores CDN Edge, lo que permite entregar el contenido lo más rápido posible a los usuarios.
- Precio. Un requisito fundamental del proyecto es minimizar costes, Firebase y su plan Spark gratuito ofrece unos límites bastante amplios que abarcan de sobra con las necesidades de la aplicación hoy en día. Además, en caso de que en un futuro la aplicación creciera, su plan Flame de 25$ al mes sería más que suficiente. Los servicios del plan Spark pueden resumirse en:
    - 10.000 conexiones por mes
    - 1 GB de Hosting de almacenamiento y 10 GB al mes de transferencia
    - Hasta 100 conexiones simultaneas a Realtime Database
    - 1 GB de almacenamiento en Realtime Database y 10 GB al mes de descarga
    - Servicio de Firebase Cloud Messaging y Analitics sin cargo.

3.3 Base y Modelo de Datos. Realtime Database

Como se explica en el punto anterior, una de las razones para el uso de Firebase como Backend es su base de datos en tiempo real, Realtime Database. También existe otra opción dentro de Firebase, Cloud Firestore, un servicio de base de datos que permite almacenar, sincronizar y consultar los datos de una manera eficiente gracias a su organización de documentos agrupados en colecciones y sub colecciones, pero cuando se comenzó el proyecto se encontraba todavía en fase beta y su documentación era más escasa así que se optó por Realtime Database.

Realtime Database es un servicio que permite crear una base de datos NoSQL alojada en la nube. Los datos se almacenan en formato JSON, convirtiendo la base de datos en un gran árbol de JSON. La característica principal de Realtime Database es la sincronización de los datos en tiempo real. Cada vez que cambian los datos, se almacenan en la nube y se notifica a los dispositivos interesados que se sincronizan de manera casi inmediata. Por otro lado, permite su uso sin conexión mediante la persistencia de datos. Cuando el dispositivo se queda sin conexión el SDK de Realtime Database, utiliza la caché local del dispositivo para almacenar los datos y, cuando el dispositivo recupere la conexión, se sincroniza con la base de datos. [24]
Como ya se ha comentado, otra ventaja de este servicio es la posibilidad de crear unas normas de seguridad, que filtren el acceso a la base de datos, limitando la escritura y lectura en la misma. Así se consigue que sólo algunos usuarios puedan acceder a ciertas

partes de la base de datos. Es cierto que el hecho de que se trate de una base de datos no relacional, limita la potencia de las consultas sobre la misma, pero consigue que estas sean más simples, permitiendo así mejorar el rendimiento para explotar su característica de uso en tiempo real.
La organización de la base de datos es un objeto JSON con ocho campos:

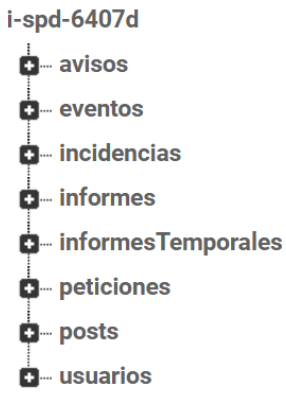


Figura 19: Base de Datos

A su vez, cada campo es un JSON, que contiene los datos en objetos también de tipo JSON con claves únicas. Por ejemplo, si desplegamos el bloque de 'avisos', veremos todos los avisos son objetos con una clave única generada por Firebase, y con los campos: "body", "date", "title", "url" y "user". Adicionalmente, si el aviso tiene comentarios, tendrá un campo "comentarios", que se trata de otro JSON, que contiene objetos también con una clave única y con los campos: "body", "date", "title", "icon", "uidUser" y "user". Este es el máximo nivel de profundidad al que llega la base de datos.

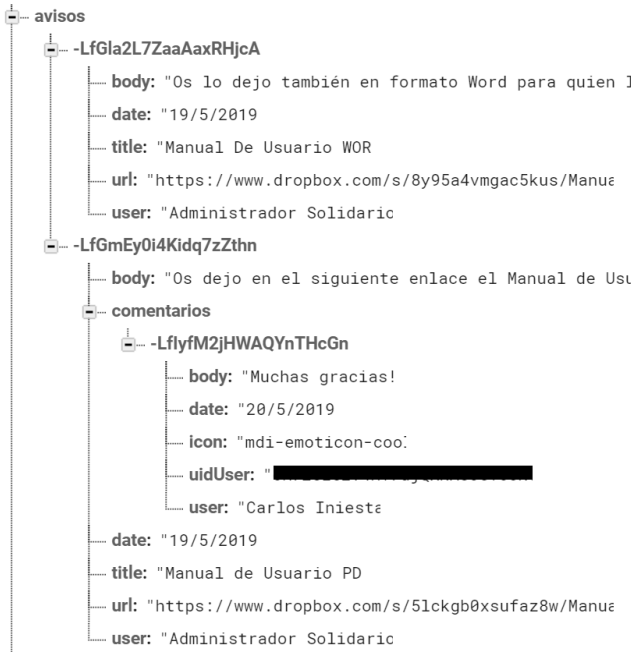


Figura 20: Avisos

A continuación, se muestra en tablas, los bloques mencionados con sus respectivos campos, donde '**key**' hace referencia a la clave del objeto JSON, que o bien es una clave

única generada por Firebase, o es el identificador único de usuario. En cualquier caso, la clave siempre es única a nivel de la colección en la que se encuentra el objeto.
La primera columna se refiere al nombre del campo, mientras que la segunda indica el tipo de dato:

AVISOS	
key	TimeStamp
body	String
date	Date
title	String
url	String
user	String
comentarios	JSON

USUARIOS	
key	Uid User
email	String
name	String
rol	String
routeName	String
routeDay	String
routeName_Day	String

PETICIONES	
key	Uid User
email	String
rol	String
name	String
routeName	String
routeDay	String
routeName_Day	String

INFORMES	
key	TimeStamp
date	Date
incident	String
nPsh	Int
nPshDor	Int
nPshH	Int
nPshM	Int
nVol	Int
routeName	String
routeDay	String
routeName_Day	String
summary	String
uidUser	Uid
user	String

INFORMES TEMPORALES	
key	Uid User
date	Date
incident	String
nPsh	Int
nPshDor	Int
nPshH	Int
nPshM	Int
nVol	Int
routeName	String
routeDay	String
routeName_Day	String
summary	String
user	String

POSTS	
key	TimeStamp
body	String
color	String
date	Date
routeName	String
routeDay	String
routeName_Day	String
title	String
url	String
userName	String
uidUser	Uid
comentarios	JSON

COMENTARIOS	
key	Stamp
body	String
date	date
icon	String
user	String
uidUser	Uid

EVENTOS	
key	TimeStamp
body	String
color	String
startDate	Date
endDate	Date
startTime	Time
endTime	Time
fullTitle	String
title	String

INCIDENCIAS	
key	TimeStamp
body	String
date	Date
state	String (A, E, C)
title	String
routeName	String
routeDay	String
routeName_Day	String
userName	String
uidUser	Uid
comentarios	JSON

4 Desarrollo

El proyecto ha pasado por varias fases, las primeras más de análisis, diseño y formación en las tecnologías que se decidieron utilizar para desarrollar la aplicación. Tras estas, comenzó la fase de desarrollo, empezando con la plantilla expuesta en el apartado 3.2.3 y siguiendo los patrones explicados.

El código de la aplicación se encuentra en un repositorio privado de github. Si se desea obtener acceso para poder seguir la explicación del desarrollo a nivel de código, puede solicitarse escribiendo un correo a: carlos.iniesta@estudiante.uam.es.

Si observamos el histórico de commits realizados, un total de 65, podemos ver claramente las distintas fases que ha ido teniendo el desarrollo del proyecto. Comenzando el 4 de noviembre de 2018, donde se hacía de media un commit por semana, puesto que la soltura con el lenguaje y el entorno era limitada y las funcionalidades más largas de desarrollar. A medida que se consiguió familiarizar con las tecnologías, la frecuencia de commits creció, además de que los cambios no eran tan radicales, ya que la base de la aplicación estaba creada y tan sólo se iban perfilando y añadiendo detalles. Por último, en mayo, se aceleró para tener lo más estable y completa posible la aplicación para las pruebas reales, que comenzaron el 20 de mayo, semana donde se realizaron los últimos commits, corrigiendo las incidencias que habían ido surgiendo.

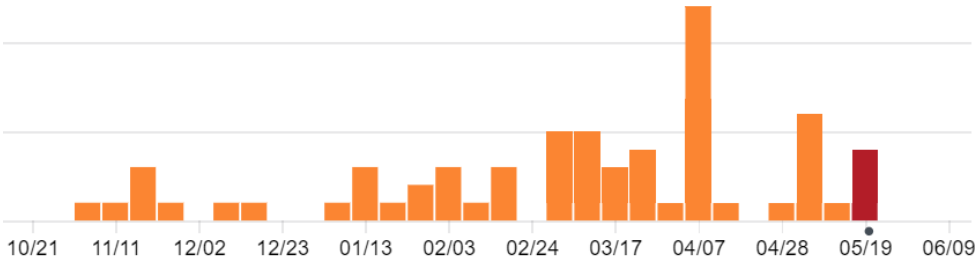


Figura 21: Histórico de Commits

Destacar que durante el desarrollo de la aplicación, se utilizó de manera reiterativa el componente "<v-card>" de Vuetify, con la finalidad de que los usuarios no tuvieran que familiarizarse con muchos componentes diferentes y conseguir así un diseño simple, coherente, unificado e intuitivo. Este componente es el utilizado para los muros de post, informes e incidencias, así como para los formularios, la lista de usuarios y la propia página de Login.

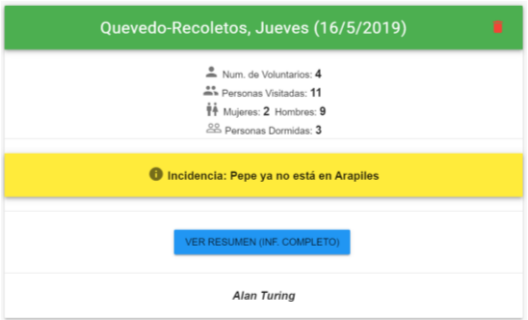


Figura 22: Ejemplo <v-card> en la App

Para explicar el desarrollo completo de la aplicación, se ha decidido dividir en bloques funcionales, puesto que explicar el desarrollo página por página sería excesivamente extenso y repetitivo.

4.1 Login

La autenticación en la aplicación se realiza a través de Google como proveedor de identidad. Es decir, los usuarios se identifican con su cuenta de Google mediante un Pop Up, y en caso de que la autenticación sea correcta, se recibe un objeto "auth", que contiene la información del usuario: su nombre, correo, uid, imagen de perfil si tiene, etc.
Para abrir el Pop Up, se debe clicar en la "G" de Google de la página de Login, que es la que se abre por defecto cuando se accede a la URL "cantarranas.es" o, se abre la aplicación.

Figura 23: Página de Login

La función que muestra el Pop Up es propia de Firebase, y se llama desde el método *"conectar()"* :

```
var provider = new firebase.auth.GoogleAuthProvider();
conectar: function() {
  firebase
    .auth()
    .signInWithPopup(provider)
    .catch(function(error) {
      console.error("Error haciendo logIn: ", error);
    });
}
```

Figura 24: signInWithPopup()

Cuando el usuario se identifica, se conoce gracias a que se ha establecido un observador sobre el objeto 'Auth' que, cuando detecta un cambio, guarda los datos de 'Auth' en el objeto *"user"* y lo almacena en el 'store' de Vuex. Esto se realiza mediante una promesa y la función de Firebase *"onAuthStateChanged()":*

```
return new Promise((resolve, reject) => {
  firebase.auth().onAuthStateChanged(function(user) {
    store.commit("setUser", user);
    resolve();
  });
});
```

Figura 25: onAuthStateChanged()

A su vez se establece otro observador, sobre la variable "*user*" de Vuex, en la página de Login. Cuando este objeto tiene contenido, quiere decir que el usuario se ha autenticado y sus datos se han guardado en el 'store', por lo que navegamos a la página *"/access"*.

```
watch: {
  user: function(user) {
    if (user) {
      console.warn("conectado: ", user);
      this.$router.replace("/access");
    } else {
      this.$router.replace("/");
    }
  }
},
```

Figura 26: observador en *"user"*

La página *"/access"* visualmente es un simple 'loader', pero es la capa intermedia entre el Login y la aplicación, donde se comprueba si el usuario está registrado en la misma. En caso de estar registrado, se obtiene su perfil desde Firebase, compuesto por su nombre en la aplicación, su ruta, su día de ruta, el correo asociado y su rol. A continuación, se guarda en la variable *"userRol"*, que también se almacena en el 'store' de Vuex para que sea accesible desde cualquier otra página de la aplicación. Seguidamente, se le concede acceso a la aplicación, dirigiéndolo a la página principal (*"/home"*), con el acceso habilitado a las páginas correspondientes en función de su rol.
En caso de que no tuviera un perfil en la aplicación, se guarda un objeto *"userRol"* anónimo, para que pueda acceder a las funciones básicas públicas de la aplicación, y se le redirige a la página *"/newUser"*, donde si es voluntario podrá solicitar el acceso a la aplicación mediante un sencillo formulario que será enviado a los Administradores.
Toda esta lógica se realiza con dos funciones, la primera se llama desde el evento *"created()"* (que se ejecuta cuando la página se ha creado), y consulta si el usuario autenticado tiene perfil con rol en el bloque *"/usuarios"* de la base de datos mediante su 'uid'. La respuesta se pasa a la función *"setUsu()"*, que almacenará en el 'store' dicho resultado o el objeto *"anonymous"*, y redirigirá al usuario a la página correspondiente.

```
created() {
  this.consultarRol();
},
methods: {
  consultarRol() {
    var userUid = firebase.auth().currentUser.uid;
    return firebase
      .database()
      .ref("/usuarios/" + userUid)
      .once("value", snapshot => this.setUsu(snapshot.val()));
  },
  setUsu(usu) {
    if (usu) {
      this.$store.commit("setRol", usu);
      this.$router.replace("/home");
    } else {
      this.$store.commit("setRol", this.anonymous);
      this.$router.replace("/newUser");
    }
  }
}
```

Figura 27: Página *"/access*

4.1.1 Roles de Usuario y Páginas

Uno de los requisitos definidos para la aplicación era la clasificación de los usuarios en función de su rol y la limitación de acceso a las distintas páginas en función del mismo. El rol por tanto va asociado al usuario desde que se crea su perfil en la base de datos, y puede ser editado por un Administrador. Los roles que existen y las páginas a las que tienen acceso son:

- **Invitado/Anónimo**: Pagina de Bienvenida (*"/home"*), Calendario (*"/calendar"*), Mi Cuenta (*"/myAccount"*) y Contacta con Nosotros (*"/contactUs"*).

- **Voluntario**: Invitado/Anónimo + Muro de Cantarranas (*"/posts"*), Nuevo Informe de Ruta (*"/newInform"*) y Muro de Informes de Ruta (*"/informs"*).

- **Coordinador**: Voluntario + Estado de mi Ruta (*"routeState"*), Nueva Incidencia/Seguimiento (*"/newIncident"*), Seguimientos e Incidencias (*"/incidents"*), Mapa de PSH (*"/maps"*).

- **Administrador**: Coordinador + Nuevo Aviso ("/newAd"), Gráficos ("/graphics") y Administrar Usuarios ("/adminUsers").

Esta división de páginas puede verse reflejada en el propio 'Navigation Drawer' que permite el acceso a las diferentes páginas y que muestra tan sólo aquellas a las que puedes acceder. Esa lógica se realiza desde el layout *"default"*, que es el que se mantiene en toda la aplicación, a excepción de las páginas de Login y Acceso. En la Figura 28, se muestra como se vería el 'Navigation Drawer' con el rol de Administrador.

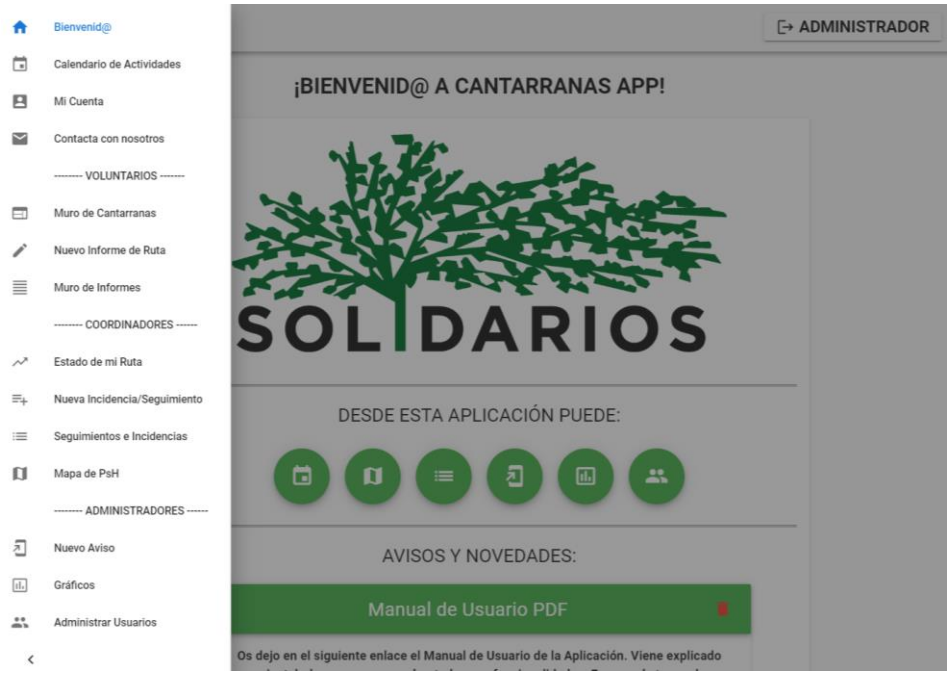


Figura 28: Páginas Administrador

4.1.2 Logout

Es importante destacar, que si se desea también se puede hacer cerrar la sesión, por si por ejemplo el usuario desea autenticarse con otra cuenta. Para ello está el botón que se encuentra en la parte superior derecha, en la 'toolbar'. Este es parte del layout "default"

común a toda la aplicación, contiene un icono de Logout y el nombre del usuario en sesión (en la Figura 28, es 'Administrador'). Al clicarlo, se llama al método *"disconnect ()"*, que mostrará un 'alert' para confirmar si es lo que el usuario desea hacer, y en caso afirmativo ejecuta la función de Firebase *"signOut()"* y redirige al usuario de nuevo a la página de Login.

```
disconnect: function() {
  if (confirm("¿Estás segur@ de que deseas cerrar sesión?")) {
    this.$router.replace("/");
    firebase
      .auth()
      .signOut()
      .catch(function() {
        console.error("Error haciendo logOut: ", error);
      })
      .then(data => {
        this.$router.replace("/");
      });
  }
},
```

Figura 29: signOut

4.2 Calendario

Una de las funcionalidades que se definieron para la aplicación durante las reuniones con la ONG, fue un calendario donde los Administradores pudieran añadir los eventos de la organización, así como los turnos de limpieza y días de cierre del local.

Para ello, se buscó un componente calendario simple de una librería externa, ya que en la librería de Vuetify no existía en ese momento (en la versión 1.5, publicada en Febrero de 2019 sacaron el componente v-calendar, cuya incorporación es una de las tareas a futuro contempladas en el apartado 6.2). Se buscó en la página https://awesome-vue.js.org, donde se encuentran enlaces a numerosos proyectos y componentes realizados con Vue y de libre disposición. Entre los calendarios existentes, se realizaron pruebas online, editando con https://codesandbox.io/, y finalmente se decidió utilizar el proyecto: vue2-calendar (https://github.com/Trekels/vue2-calendar).

La principal razón de escoger este calendario fue su simplicidad de uso y la documentación que aportaba para su integración. Se instaló su paquete a través de npm con el comando: *"$npm install vue2-simple-calendar"* y tan sólo hubo que importar el paquete y desarrollar la lógica para crear nuevos eventos, que tenían que incluir los siguientes campos para pintarse en el calendario: 'title' (String), 'start' (Date) y 'end' (Date). Además de estos campos, se añadieron más para completar la utilidad, como descripción del evento, horario de comienzo y final del mismo y el color. Por otro lado, se hizo una distinción con el título, ya que se observó que ante títulos largos, se comportaba mal en pantallas pequeñas (menos de 5 pulgadas), por lo que el campo 'title', que se muestra en el calendario, tiene un máximo de 3 caracteres, mientras que el campo 'fullTitle', que se muestra cuando se accede a la información del evento puede tener hasta 17 caracteres.

El calendario funciona a través de eventos, que se lanzan con la directiva "eventBus". Cada vez que se clica en la casilla de un día se lanza el evento "day-clicked", con la fecha del día seleccionado. Se abre entonces un Pop Up a modo de formulario, para escribir la información del evento, con la fecha ya precargada. Cuando se completa y se da a crear, la información se guarda en la base de datos, en la rama "/eventos". Cabe destacar, que sólo los administradores pueden crear nuevos eventos, por tanto los demás roles no tienen permiso de escritura en esa rama de la base de datos y además, no se les muestra el Pop Up de creación de evento al clicar en un día. En este formulario, existe la posibilidad de editar el color del evento, para que el uso del calendario sea más eficiente, además también

permite crear de una forma rápida eventos de turnos de limpieza, que se crean de color amarillo, con descripción y horario prefijados.
De manera similar, es el funcionamiento para mostrar información de los eventos del calendario, cuando se clica sobre uno, se lanza el evento "event-clicked", que contiene la información del evento que se carga en otra ventana Pop Up. Todos los usuarios autenticados tienen permiso de lectura, pero a los Administradores se les habilita además la opción de eliminar el evento.

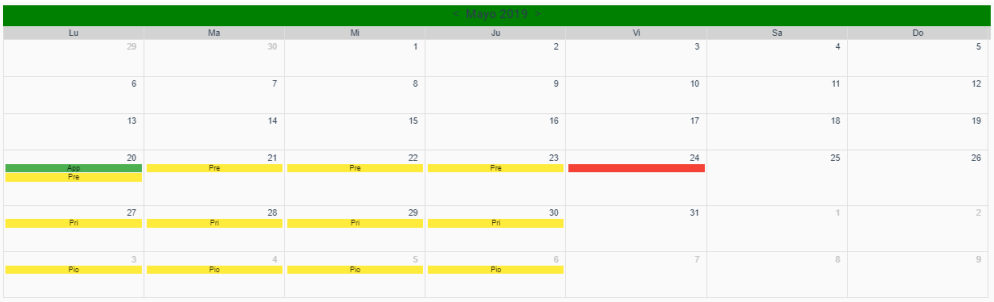


Figura 30: Calendario

4.3 Formularios

Un elemento común de la mayoría de funcionalidades que tiene la aplicación son los formularios. Estos se utilizan para la petición del alta la primera vez que se accede, para escribir los avisos en la pantalla de inicio, crear eventos, publicar post, escribir informes de ruta e incidencias, así como para escribir comentarios y editar las cuentas de los usuarios.
Todos ellos están desarrollados sobre el componente "<v-card>" de Vuetify, y haciendo uso del componente interno "<v-form>" y su propiedad *"rules"* para validar campos.
Para los campos de los formularios, se han utilizado los siguientes componentes:

- **"<v-text-field>":** para permitir introducir texto reducido como títulos, nombres o URL's. Además se utilizaba también para números y fechas, indicándolo en la propiedad "*type*" como "*number*" o "*date*", respectivamente.
- **"<v-textarea>":** textos más extensos como los resúmenes de los informes, cuerpo de las incidencias y avisos, así como la descripción de los eventos,
- **"<v-select>":** para cuando se debía elegir entre unas opciones limitadas, como el día de la ruta, el rol o el estado de las incidencias.
- **"<v-item-group>":** se utiliza en los formularios donde se permite seleccionar el color de la publicación. Dentro de este componente, se realiza un bucle 'for' ("<v-for>") con los distintos colores que se muestran en pequeñas cajas del color correspondiente.
- **"<v-btn>":** utilizado para los botones que publicaban o enviaban el formulario, con la propiedad "*type=submit*", accionaban el evento "*submit.prevent*" que a su vez llamaba a la función '*validateForm()*', que tras validar que se habían completado todos los campos requeridos, llamaba a la función que conectaba con Firebase y guardaba los datos en el lugar correcto.

Un ejemplo de un campo de un formulario podría ser el título que tienes que escribir cuando quieres publicar un nuevo post en el Muro de Cantarranas. Es un texto corto, al que le añadimos el icono *"title"* de Vuetify, cuyo valor se guarda en la variable *"post.title",* tiene un texto indicativo *"label"*. Además se trata de un campo requerido, posee un contador de caracteres y su longitud máxima es de 50 caracteres:

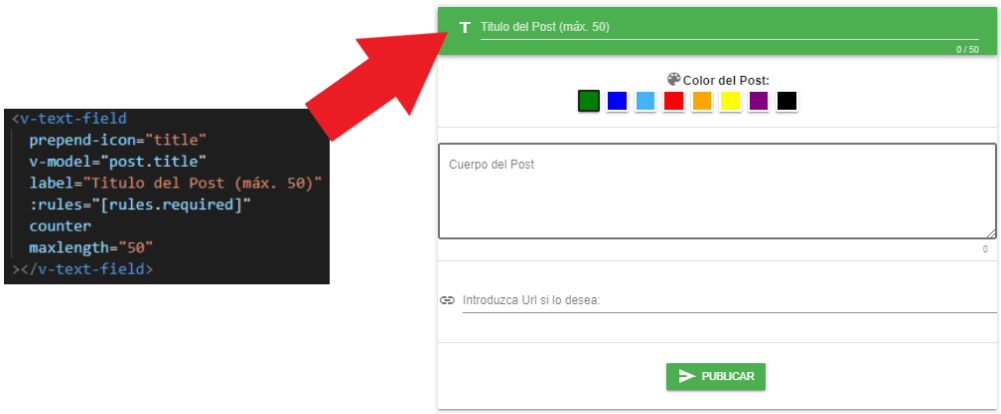


Figura 31: Formulario Post

4.4 Listas

Las listas o muros son un elemento recurrente en la aplicación. Encontramos el muro de avisos y novedades en la pantalla de inicio, el muro de post llamado 'Muro de Cantarranas', el de los Informes de Ruta y las Incidencias, así como las listas de peticiones de alta y de usuarios a las que sólo pueden acceder los Administradores.

Los componentes que hacen la función de muro o lista, tienen en común que realizan una llamada a Firebase en el evento *"created()"*, es decir, cuando se crea el componente. Esta petición recibe de la base de datos el número de objetos que queramos de la rama que deseamos consultar, el resultado se pasa por argumento a una función encargada de cargar los datos obtenidos en una variable tipo array. Mediante la directiva "<v-for>", se recorren los elementos de dicho array y se pinta cada objeto en un componente "<v-card>", con una estructura similar al formulario en el que se escribieron.

Un ejemplo puede ser el componente *"Ads.vue"*, que corresponde al muro de avisos y novedades que se encuentra en la página de bienvenida. Podemos observar como en el *"created()"* se consultan los últimos 5 avisos en la base de datos y su resultado se carga en la variable *"avisos"*, para recorrer dicha variable pintando los objetos en "<v-card>":

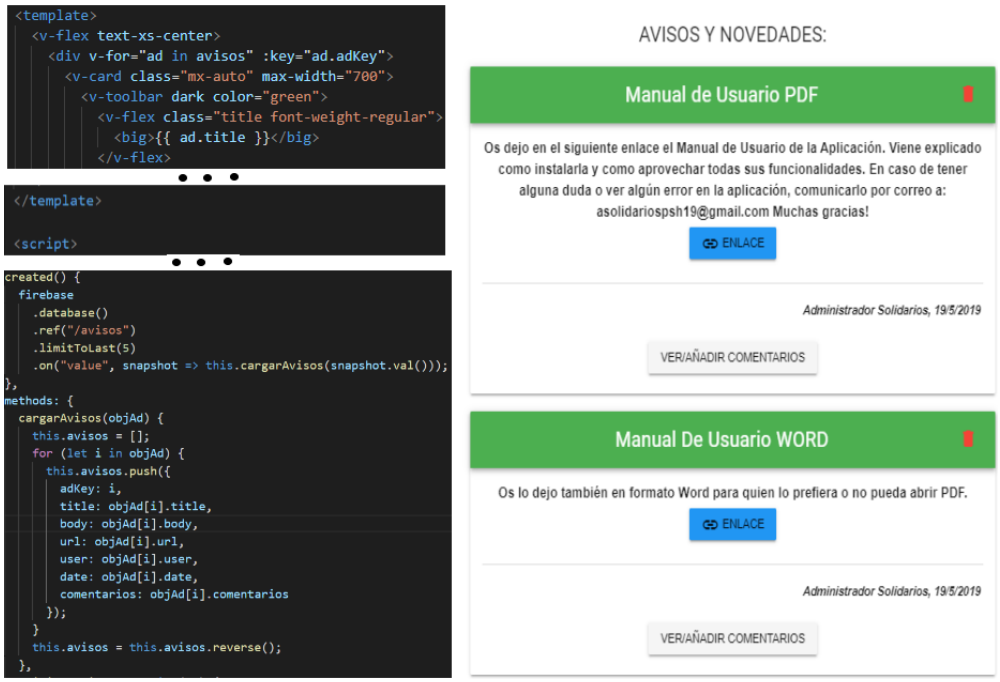


Figura 32: Muro y lógica de Avisos

Dentro de las listas o muros, se encuentran algunos elementos importantes como son:

- **Botón 'Enlace':** Estos botones aparecen cuando un aviso o post, tiene una URL. Para no poner toda la dirección, se añade un botón con el nombre 'Enlace' y el icono correspondiente, que redirige a la URL que se escribió en el formulario.
- **Filtros:** Son una parte fundamental de algunos muros, especialmente en los de informes, incidencias y usuarios. Para usar estos filtros, hay un botón de color azul con el nombre 'Filtros', que los despliega y permite filtrar por los distintos campos que tengan los objetos del muro en concreto. Además, permite seleccionar el número de objetos máximo, ordenados desde el más moderno, que se desean obtener. El funcionamiento de estos filtros es básicamente una consulta a la base de datos Realtime Database. Haciendo uso de la función *"orderByChild()"* de Firebase, filtramos por el campo que deseemos en la colección correspondiente. El resultado de la consulta se guarda en la misma variable tipo array, donde se almacenaba la consulta inicial y, gracias a la reactividad de Vue, la página se actualiza con los nuevos datos.

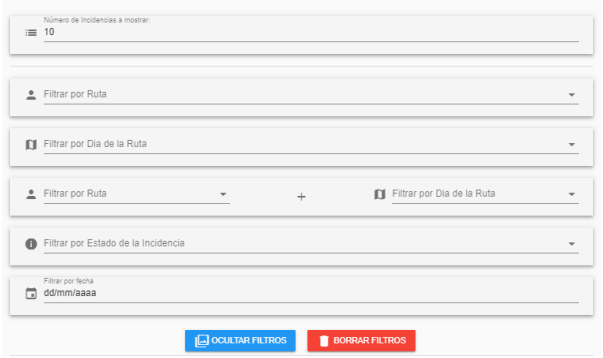


Figura 33: Filtros de Incidencias

- **Buscadores:** Es otro elementos que tienen en común varios muros. Estos buscadores filtran a nivel local, no mediante una llamada a Firebase, los objetos de la variable de tipo array que contiene los datos. Se filtran aquellos que contienen los caracteres que se escriben en el buscador en el campo correspondiente, gracias al uso de las funciones *"filter"* e *"include"* propias de JavaScript. En incidencias y posts filtra por el título, en informes por el resumen y en usuarios por el nombre de usuario.

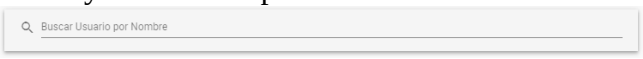


Figura 34: Buscador de Usuarios

- **Comentarios:** Por último, cabe destacar la funcionalidad de poder escribir comentarios sobre los elementos de alguna lista, como los avisos, incidencias y post. Para poder visualizar y escribir comentarios, se debe seleccionar en el botón 'Ver/Añadir Comentarios' del elemento deseado. Se desplegarán entonces los comentarios del mismo, y se mostrará un cuadro de texto donde poder añadir uno nuevo. Además es posible seleccionar el emoticono que acompaña al comentario.

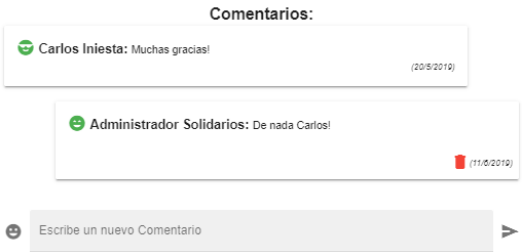


Figura 35: Comentarios

4.5 Gráficos

Uno de los beneficios y objetivos de realizar los informes de ruta a través de un formulario, es poder explotar los datos recogidos en ellos, generando gráficos que permitan consultar el estado y evolución de las rutas de una manera ágil y visual. Existe la posibilidad de generar tres gráficos: el primero muestra el número de voluntarios que han ido a cada ruta, el segundo el número de personas sin hogar que se visitaron y el tercero el número de personas dormidas que se encontraron. Todos ellos se muestran siempre desde el dato más antiguo al más moderno, de izquierda a derecha. Además, todos comienzan en cero, para que se comparen sobre la misma base.

Para la realización de estos gráficos se utilizó el componente de Vuetify "<v-sparkline>", añadido en la versión 1.4 de diciembre de 2018. Este componente, recibe en su propiedad "*value*", un vector de números y dibuja el gráfico. Además, posee una serie de propiedades para editar el propio gráfico, como "*labels*" para mostrar el número en el eje de abscisas, "*smooth*" el grado de suavidad de la gráfica, "*widht*" para el ancho, etc. Para editar estas propiedades, se hay una serie de controles en la parte inferior de la página (Figura 37).

Los gráficos pueden ser utilizados por los Coordinadores, en la sección 'Estado de mi Ruta', donde se obtiene del objeto *"userRol"*, la ruta y día de ruta de la que son coordinadores y se realiza el gráfico con los últimos 20 informes publicados de esa ruta, aunque ese número puede ser editado por el usuario. Además, existe un apartado específico de gráficos de acceso exclusivo para los Administradores. En esta página se generan por defecto los tres gráficos cogiendo los últimos 20 informes publicados sin importar la ruta ni día de ruta. Pero existe la posibilidad de filtrar, de igual manera que en los Muros, pudiendo obtener gráficos para una ruta específica, día y ruta, fecha, etc.

El funcionamiento, al igual que con las listas, comienza con una consulta a los informes de la base de datos, pero de todos los campos de cada objeto, esta vez tan sólo se guardan el número de voluntarios, el de personas visitadas y el de personas dormidas. Se almacenan en tres vectores separados, que se pasan por la propiedad *"values"* al correspondiente componente "<v-sparkline>".

Figura 36: Gráfico

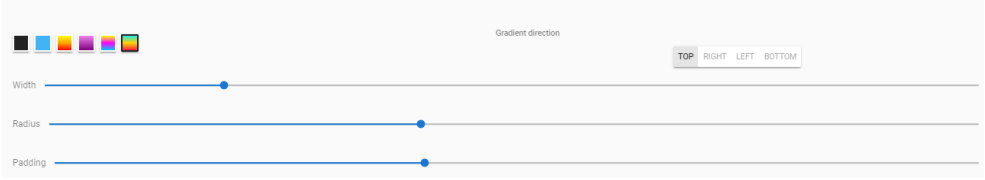


Figura 37: Editor de Gráficos

4.6 Seguridad

A pesar de que los datos que se manejan en la aplicación no son especialmente delicados al no tener un especial valor fuera del programa, y además de no manejar contraseñas de usuarios, la seguridad es una parte importante en el proyecto, en la que se ha trabajado por dos vías.

4.6.1 Seguridad en Frontend

El primer camino fue asegurar los accesos a las páginas en función del rol de usuario. Cuando un usuario se auténtica, tanto en el "Navigation View" como en la página de Bienvenida, sólo aparecen las opciones para navegar a las paginas a las que el usuario tiene permiso en función de su rol, esta limitación es fácilmente esquivable, por ejemplo escribiendo a mano la URL en el navegador si se accede a la aplicación a través de la web.
Para evitarlo, se ha asegurado el acceso a las páginas restringidas a un cierto rol mediante una funcionalidad propia de Nuxt.js, los middlewares. Estos son archivos de JavaScript, que se guardan en la carpeta homónima, y se importan en las páginas donde se desean usar. Se ejecutan justo antes de cargar la página en donde son importados, y su función es obtener el rol del usuario conectado y comprobar si es el adecuado para acceder a la página en cuestión, en caso contrario muestra una alerta y redirige al usuario a la página de inicio.
El siguiente ejemplo, muestra el middleware que tienen las páginas exclusivas de los Administradores:

```
export default function({ store, redirect, route }) {
  const home = "/home";
  const access = "/access";
  //Si no tiene rol redirigimos a /access
  if (store.getters.getRol === null) {
    redirect(access);
  }
  //Si tiene rol vemos si es el correcto para acceder
  if (store.getters.getRol !== "Administrador") {
    alert("A esta página sólo pueden acceder los Administradores");
    redirect(home);
  }
}
```

Figura 38: Middleware Administradores

4.6.2 Seguridad en Backend

Cómo es evidente, no basta con limitar los accesos a través de la aplicación, es necesario proteger los datos al nivel de la propia base de datos. Para ello, como se comentaba en el apartado de Diseño, Firebase provee de una herramienta muy útil para Realtime Database, que consiste en unas reglas personalizadas para controlar los permisos de lectura y escritura en tu base de datos. Si observamos la arquitectura básica de la aplicación (Figura 11), podemos ver como toda petición a la base de datos pasa por estas reglas, que evalúan la petición y permiten/deniegan el acceso en función del usuario que la realiza. Este desarrollo se explica con más detalle en el Anexo B: Reglas de Seguridad de Firebase.

De manera adicional, también se ha protegido el acceso a la consola de Firebase desde donde se maneja la base de datos y el hosting de la aplicación, mediante una cuenta de Google con verificación en dos pasos y una contraseña compleja que se va cambiando.

5 Pruebas y resultados

Un aspecto fundamental de todo proyecto de desarrollo que aspira a ser funcional son las pruebas. Entre los objetivos del proyecto en esta primera fase estaba poder tener la aplicación en fase beta, siendo utilizada por usuarios reales para poder testear sus funcionalidades y recoger las sensaciones del usuario final. Por otro lado, también se han realizado pruebas durante el desarrollo de la misma, asegurando que se iba construyendo una aplicación estable y segura.

5.1 Pruebas durante el desarrollo

A medida que se ha ido construyendo la aplicación, se realizaban pruebas con cada nueva funcionalidad. Una de las grandes ventajas de desarrollar aplicaciones web, es poder desarrollar al mismo tiempo que la aplicación se encuentra en un servidor local, abierta desde el navegador y actualizándose cada vez que se guarda el código. Esto, permite ir probando cada elemento o funcionalidad nueva al mismo tiempo que se desarrolla, realizando test unitarios y poco a poco test de integración con los demás elementos. Un gran apoyo durante los desarrollos ha sido la herramienta "Vue DevTools", una extensión de Chrome, que permite depurar las aplicaciones realizadas con Vue, teniendo por ejemplo acceso al 'store' de Vuex. Además, también se ha hecho uso constante de las herramientas de desarrolladores de Chrome, que permite inspeccionar la aplicación web, tener acceso a la consola y poder simular como se vería la aplicación para distintos tamaños de pantalla, una práctica fundamental para el desarrollo de aplicaciones multiplataforma.

A parte de las pruebas que se iban realizando de manera entrelazada al desarrollo, se realizaban pruebas específicas antes de subir cada commit y también previamente a realizar un despliegue. Para estas pruebas se escribía una lista de las nuevas funcionalidades y de los flujos que estaban implicados en las mismas. Estos se realizaban con distintos usuarios y roles, y no se realizaba el commit o el despliegue hasta no pasarlos todos.

Por último, tras cada despliegue se realizaban los mismos flujos en varios dispositivos, que abarcaran las distintas plataformas y tamaños de pantalla, comprobando también así que se cumplían los requisitos de que fuera multiplataforma e intuitiva sin importar el tamaño de pantalla. También se iba comprobando el espacio que ocupaba la aplicación. Para ello se contó con los siguientes dispositivos en los que se realizaban las pruebas:

- Xiaomi Redmi Note 5 de 5.99 pulgadas (dispositivo Android y de pantalla grande).
- Iphone 5s de 4 pulgadas (dispositivo IOS y de pantalla pequeña).
- Surface 3 pro de 12 pulgadas (ordenador Windows pantalla mediana).
- Acer Extensa 2540 de 15.6 pulgadas (ordenador Linux con pantalla grande).
- Apple MacBook Air de 13 pulgadas (ordenador Mac OS con pantalla mediana)

5.2 Pruebas reales

Como ya se ha comentado, uno de los objetivos marcados al comenzar el proyecto era tener la aplicación en pruebas con los usuarios reales al final del curso, de cara a revisar su utilidad, detectar errores, recoger propuestas y principalmente, revisar si cumplía con los requisitos definidos, tanto funcionales como no.
Por ello, el lunes 20 de mayo se comenzó a utilizar la aplicación, se realizó previamente un manual de usuario que fue enviado a todos los voluntarios y, de manera complementaria, el alumno fue durante la semana al local de la organización, para dar soporte a los usuarios

antes de que salieran de ruta, ayudar con la instalación y enseñar el uso básico de la aplicación.
Esta fase de pruebas se realizó intensamente durante esa semana, y se ha mantenido hasta la actualidad con los voluntarios que desean seguir familiarizándose con la aplicación y testeando la misma para sugerir mejoras. Podemos comprobar esto utilizando las herramientas de análisis de uso de nuestra aplicación que proporciona Firebase, por ejemplo, consultando las descargas en MB desde el hosting de la aplicación en el mes de mayo.

Figura 39: Hosting Downloads

El resultado de estas pruebas fue realmente satisfactorio, se pudo comprobar que se habían cumplido los requisitos funcionales y no funcionales estipulados. La aplicación sirvió como herramienta para subir los resúmenes de ruta, creando un repositorio de los mismos y pudiendo generar gráficos con los datos que incluyen. Por otro lado, se comprobó la utilidad del calendario para los turnos de limpieza y aviso de eventos, además de la posibilidad de realizar un registro de los voluntarios activos. En definitiva, se logró conectar a los voluntarios de la organización y facilitar las tareas de organización.
Respecto a los no funcionales, se pudo testear de la mejor manera la característica multiplataforma de la aplicación, ya que durante la primera semana fue utilizada por más de 50 voluntarios con dispositivos móviles de todo tipo, ordenadores y tabletas. Por otro lado, se comprobó que se había cumplido el criterio de minimizar el espacio que ocupa la aplicación, permitiendo su instalación y uso en teléfonos con poca memoria disponible (<50 MB). Además, se reafirmó y se valoró el requisito logrado de seguridad, tanto en la autenticación al utilizar los servicios de Google, como en el aspecto del acceso limitado a las funcionalidades para cada rol de usuario.
Por último, se pudo observar que también se cumplió el objetivo de que fuera intuitiva, ya que sin mucha ayuda, usuarios de todos los perfiles supieron manejarse con cierta soltura, siendo capaces de realizar el flujo funcional principal de la aplicación, publicar y consultar informes de ruta.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Tras la finalización del proyecto, podemos concluir que el resultado ha sido realmente positivo. La aplicación desarrollada cumple con los requisitos, tanto funcionales como técnicos, marcados en la fase de análisis. Además, se ha alcanzado el objetivo de tener la aplicación en fase beta, siendo utilizada por usuarios reales, con el fin de comenzar su uso de manera completa en septiembre de 2019.

Se ha conseguido el objetivo de desarrollar una aplicación multiplataforma, ligera (actualmente ocupa 274 KB, que llegan a unos 350 con los datos en caché), segura, intuitiva y todo ello a coste cero.

El desarrollo de este proyecto ha permitido al alumno investigar y analizar las diferentes alternativas existentes para realizar aplicaciones multiplataforma, así como las diferentes posibilidades para dotar a estas de Backend.

Por otro lado, se han podido elegir las tecnologías más óptimas para el proyecto entre las analizadas y formarse para ello. Aprendiendo a programar aplicaciones web progresivas, con el framework de JavaScript Vue, apoyado en Nuxt y en la librería Vuetify.

Además, se ha adquirido conocimiento en la utilización de MBaaS con el uso de Firebase, especialmente útil ha sido aprender a construir una base de datos en tiempo real e interactuar con ella, gracias a Realtime Database.

A parte de los conocimientos técnicos adquiridos, cabe destacar que se ha realizado un proyecto completo, lo que ha permitido al alumno realizar las fases de análisis y captura de requisitos para la posterior elección de las tecnologías más adecuadas, todo ello en conjunto con el "cliente", que en este caso es la ONG Solidarios Para el Desarrollo. Para terminar implantando la aplicación y formando a los usuarios para su uso.

El desarrollo de la aplicación Cantarranas tal y como se encuentra ahora, es sólo la primera piedra. Ya se han planteado mejoras y nuevas funcionalidades que se irán desarrollando en función de lo exitoso que resulte su uso a partir de septiembre de 2019

El proyecto ya ha logrado mejorar la calidad de la acción social que realiza el programa de personas sin hogar de Solidarios, tratando de optimizar su organización, conectar a sus voluntarios y explotar los datos que recogen, con el fin de llegar a más personas y de la mejor manera posible. Prueba de ello es que cuenta ya con más de 60 usuarios, con previsión de que para septiembre sean más de 100.

Como conclusión final, me gustaría destacar la tecnología como herramienta capaz de generar un impacto en la sociedad y, utilizarla con un enfoque solidario es fundamental para construir un mundo mejor y más justo.

6.2 Trabajo futuro

La aplicación Cantarranas, objeto central de este proyecto, se encuentra en su primera versión. Cuando se trató la idea del proyecto con los trabajadores de la ONG, se habló de que esto fuera un proyecto a futuro, y en este primer año intentar abarcar las funcionalidades más básicas y urgente. Si se van cubriendo dichas funcionalidades de manera correcta, se tratará de ir ampliando las posibilidades de la aplicación y mejorando sus características.

La lista de trabajo a futuro es extensa, algunas de las ideas han surgido directamente de los usuarios tras las pruebas, otras no se han podido realizar por falta de tiempo o por que no estaba a punto la tecnología en cuestión. Las más importantes son:

- Refactorización y mantenimiento del código. Aunque se ha tratado de realizar durante el desarrollo, y especialmente antes de fase de pruebas reales, todavía se pueden optimizar y simplificar ciertas partes del código. Esta tarea se pretende abordar desde este mismo verano, y se mantendrá el tiempo necesario.
- Login con más proveedores. A pesar de que en un comienzo parecía suficiente utilizar el proveedor de Google, en las pruebas varios usuarios han pedido poder utilizar otros servicios de autenticación. Una solución sería utilizar el componente de Firebase "FirebaseUI Auth", recomendado además para PWA. Este, aparte de Google, permite autenticarse con Facebook, Twitter, Email y teléfono.
- Calendario de Vuetify. Como se comentó en el apartado correspondiente (4.2), se utilizó un componente de calendario externo a Vuetify, ya que no existía dicho componente en la librería. Desde febrero existe uno y, con el fin de que todos los componentes de la aplicación provengan de la misma librería, convendría sustituir el componente actual de calendario por el de Vuetify.
- Migración a Firestore. Como ya se ha explicado, se descartó su uso porque al comienzo del proyecto se encontraba en fase beta, y se prefirió optar por una base de datos más asentada como Realtime Database. Actualmente ya está en una versión estable, y su organización es más eficiente y flexible, por tanto se tiene en cuenta como una mejora a futuro la migración de la base de datos a este nuevo servicio de Firebase.
- Sección para consultar los voluntarios de tu ruta. Una de las funcionalidades requeridas por los usuarios, es que los coordinadores cuenten con una sección donde poder ver todos los usuarios que pertenecen a su grupo y su información, para facilitar la organización de las rutas.
- Más gráficos. Otra posible mejora será desarrollar más variedad de gráficos, o incluso, permitir generar los gráficos de manera interactiva, por ejemplo que se pueda obtener un gráfico donde aparezcan sólo el número de mujeres visitadas cada día de ruta.
- Notificaciones Push. Una de las funcionalidades a futuro más importantes para implementar, serán las Notificaciones Push. Principalmente se desea enviar notificaciones a los usuarios cuando se publique un informe de la ruta del usuario y cuando toque el turno de limpieza a su ruta. Para ello, se podría uso del servicio FCM de Firebase y de Cloud Functions para realizar la lógica.
- Mapas. Se pretende añadir una funcionalidad, basada en el uso de la API de Google Maps, o alguna similar, y la geolocalización de los dispositivos móviles. Dicha funcionalidad consistirá en que los Coordinadores de Ruta puedan ir añadiendo "Chinchetas", de un color para cada ruta, en la ubicación donde duerme la gente sin hogar que visitan, generando así un mapa que permita optimizar la organización de las rutas y llegar a más gente.

Referencias

[1] Red FACIAM. (11 Abril 2019). Datos destacados del IX recuento de personas sin hogar - Madrid 2018 [Online]. Available: https://faciam.org/2019/04/11/ix-recuento-de-personas-sin-hogar-en-madrid-2018-datos-destacados/

[2] Solidarios Para el Desarrollo. Historia. [Online]. Available: http://www.solidarios.org.es/quienes-somos/historia/

[3] Pablo Thomas, Lisandro Delia, Leonardo Corbalán y otros. Tendencias en el desarrollo de Aplicaciones para Dispositivos Móviles. Facultad de Informática – Universidad de la Plata. ISBN 978-987-3619-27-4 XX Workshop de Investigadores en Ciencias de la Computación. (27 de Abril 2018).

[4] Apache Cordova. Architectural overview of Cordova platform. [Online]. Available: https://cordova.apache.org/docs/en/9.x/guide/overview/index.html

[5] Ionic - Cross-Platform Mobile App Development. [Online]. Available: https://ionicframework.com/
Ionic Case Study: Marketwatch. [Online]. Available: https://ionicframework.com/enterprise/resources/case-studies/marketwatch

[6] Appcelerator. Appcelerator Open Source: Titanium, Aptana and Alloy. [Online]. Available: https://www.appcelerator.org/

[7] NativeScript. Native mobile apps with Angular, Vue.js, TypeScript, JavaScript. [Online]. Available: https://www.nativescript.org/
NativeScipt Showcases. [Online]. Available: https://www.nativescript.org/showcases

[8] React Native. A framework for building native apps using React. [Online]. Available: https://facebook.github.io/react-native/

[9] React Native. Guide of React Native. Chapter 1: React Native Internals. [Online]. Available:https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html

[10] Google. Flutter. Beautiful native apps in record time. [Online]. Available: https://flutter.dev/

[11] Microsoft. Xamarin. Documentación de Xamarin. Docs.Microsoft. [Online]. Available: https://docs.microsoft.com/es-es/xamarin/
Xamarin Customers. [Online]. Available: https://dotnet.microsoft.com/apps/xamarin/customers

[12] Alex Russell. (15 de Junio 2015). Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Infrequently Noted. [Online]. Available: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

[13] Sacha Greif, Raphaël Benitte and Michael Rambeau. The State of JavaScript 2018. [Online]. Available: https://2018.stateofjs.com/

[14] Daisy Gabriela Valencia A. "Análisis de Frameworks de desarrollo de API REST y su impacto en el rendimiento de aplicaciones web con arquitectura SPA". M.S. Thesis. Universidad tecnica del Norte, Ecuador. 11 de Mayo 2018.
Apartado: 2.2.5. Desarrollo de Frontend con SPA

[15] Node.js. Foundation Node.js. [Online]. Available: https://nodejs.org/es/about/

[16] Kin Lane. "Overview Of The Backend as a Service (BaaS) Space". Mayo 2013. [Online]. Available: http://www.integrove.com/wp-content/uploads/2014/11/api-evangelist-baas-whitepaper.pdf

[17] Firebase. Pricing Plans of Firebase. [Online]. Available: https://firebase.google.com/pricing

[18] Amazon Web Services. Capa gratuita de AWS. [Online]. Available: https://aws.amazon.com/es/free/

[19] Nuxt.js. Version 2.8 of Nuxt Guide. [Online]. Available: https://nuxtjs.org/guide

[20] Vuex. "What is Vuex?" [Online]. Available: https://vuex.vuejs.org/

[21] Vuetify. Getting Stared Guide: "Why Vuetify?" [Online]. Available: https://vuetifyjs.com/en/getting-started/why-vuetify

[22] Alexandre Chopin, Sébastien Chopin y otros. Plantilla para aplicación con Vuetify y Nuxt.js. (Noviembre 2016 - Junio 2019). . [Online]. Available: https://github.com/vuetifyjs/nuxt

[23] Google. Firebase. A comprehensive mobile development platform. [Online]. Available: https://firebase.google.com/

[24] Google. Firebase Realtime Database Documentation. [Online]. Available: https://firebase.google.com/docs/database

Nota: *Debido al alto contenido de referencias de tipo web, correspondientes a las páginas oficiales de las distintas tecnologías analizadas, y a sus frecuentes actualizaciones, se han ido revisando las diferentes páginas citadas ante posibles cambios drásticos de su contenido. La última revisión de dichas referencias se ha realizado el día* ***16 de Junio de 2019.***

Glosario

CRM	Customer Relationship Management
API	Application Programming Interface
IDE	Integrated Development Environment
PWA	Progressive Web Application
SPA	Single Page Application
ONG	Organización No Gubernamental
URL	Uniform Resource Locator
HTML	HyperText Markup Language
XML	Extensible Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
ECMA	European Computer Manufacturers Association
PHP	Hypertext Pre-Processor
SQL	Structured Query Language
NPM	Node Package Manager
NASA	National Aeronautics and Space Administration
BaaS	Backend as a Service
MBaaS	Mobile Backend as a Service
AWS	Amazon Web Services
SFC	Single File Component
SSD	Solid State Drive
CDN	Content Distribution Network
JSON	JavaScript Object Notation

Anexos

A Manual de instalación

El presente Anexo, trata de enumerar los pasos necesarios para instalar la aplicación PWA "Cantarranas", objeto de este proyecto. Los pasos pueden variar en función de la versión del navegador. Para la instalación en ordenadores Mac OS, se recomienda descargar el navegador Google Chrome si no lo tiene y seguir las instrucciones del apartado 3 del presente anexo.

1. Instalar la aplicación en móvil/tablet Android

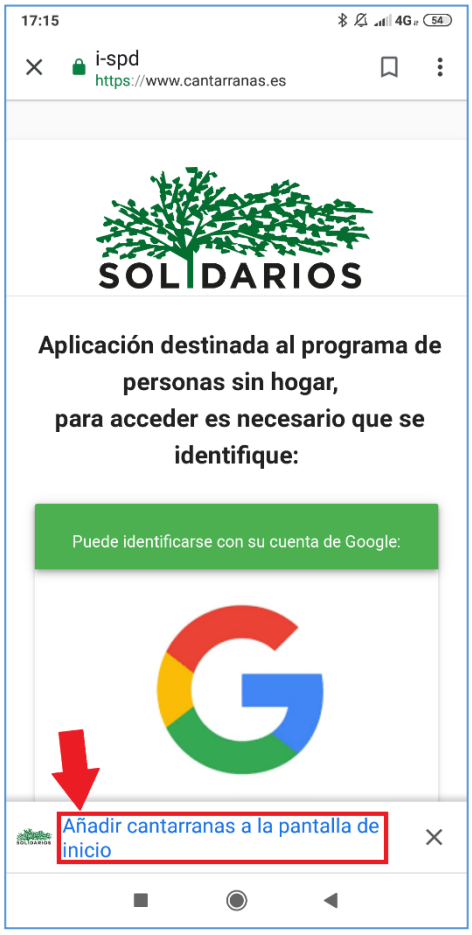
El procedimiento para instalar la aplicación en un móvil Android es el mismo que en una Tablet del mismo Sistema Operativo. Los pasos son los siguientes:

1. Accedemos a la aplicación de Chrome de nuestro dispositivo:

2. A continuación, accedemos a la web www.cantarranas.es (o simplemente "cantarranas.es").

3. Una vez en la web, si nos aparece la opción en la parte inferior de "**Añadir cantarranas a la pantalla de inicio**" (Opción A) la seleccionaremos. Si no aparece dicha opción, tendremos que seleccionar los tres puntos que aparecen en la esquina superior derecha y posteriormente pulsar "**Añadir a pantalla de inicio**" en el menú que se despliega (Opción B):

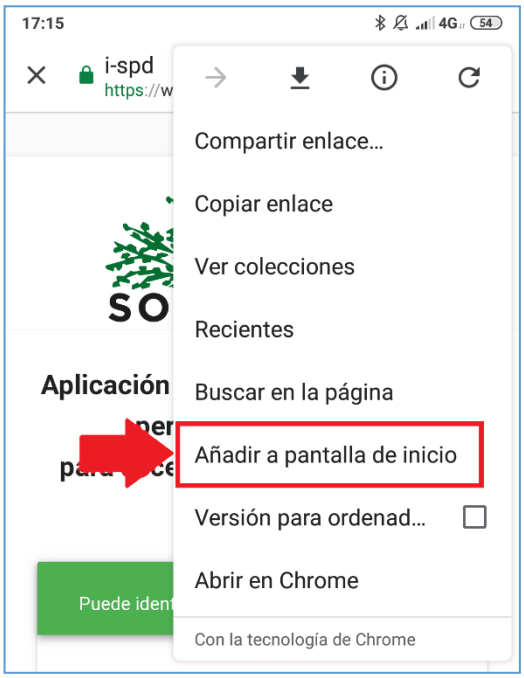
-Opción A:



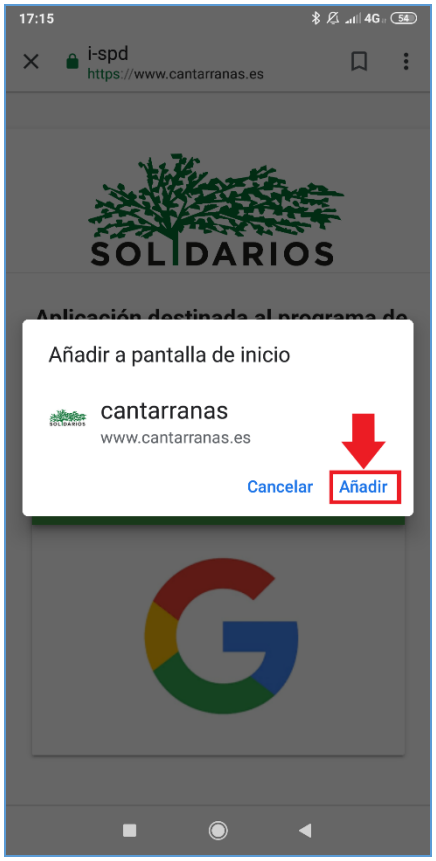
-Opción B:

- Seleccionamos los puntos de la esquina superior derecha:

- Escogemos la opción "**Añadir a pantalla de inicio**":



4. Tras realizar cualquiera de ambas opciones, nos aparecerá la siguiente pantalla donde daremos a "**Añadir**":



5. Comenzará entonces su instalación y, después de unos segundos, ya podremos disfrutar de la aplicación "Cantarranas" como una aplicación más de nuestro dispositivo con el siguiente icono:

2. Instalar la aplicación IPhone/IPad IOS

El procedimiento para instalar la aplicación “Cantarranas” en un móvil IOS (IPhone) o en una Tablet IPad es el mismo. Para ello deberán realizar los siguientes pasos:

1. Accederemos a la aplicación de Safari desde nuestro dispositivo:

2. A continuación, navegaremos a www.cantarranas.es (o simplemente “cantarranas.es”).

3. Una vez estemos en la web, seleccionaremos el icono compuesto por un cuadrado con una flecha hacia arriba, que se encuentra en la parte inferior de la pantalla:

4. Se nos desplegará un menú de opciones, donde debemos buscar y seleccionar la opción **“Añadir a la página de inicio”**: *

**(Esta opción puede no verse a simple vista y encontrarse oculta en la parte derecha)*

5. Nos aparecerá entonces una pantalla donde podemos cambiar el nombre de la aplicación y, donde tendremos que dar a “**Añadir**” en la parte superior derecha:

6. Automáticamente se instalará la aplicación y, tras unos segundos, podremos hacer uso de ella como cualquier otra aplicación de nuestro dispositivo con el siguiente icono:

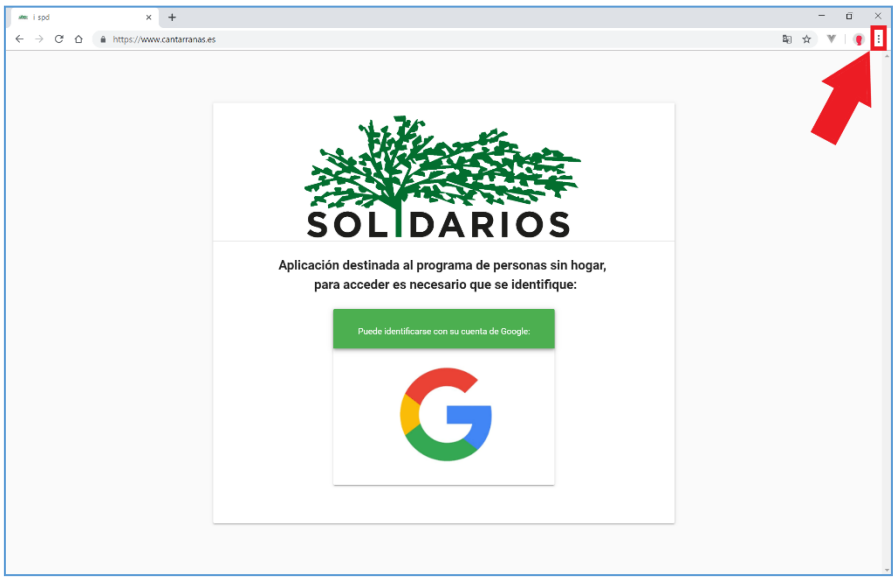
3. Instalar la aplicación en ordenador con Google Chrome

Para instalar la aplicación en un ordenador se recomienda el uso del navegador Chrome. Se realiza de manera similar en cualquier ordenador, aunque pueden existir variaciones en función de la versión del navegador. Para ello realizaremos los siguientes pasos:

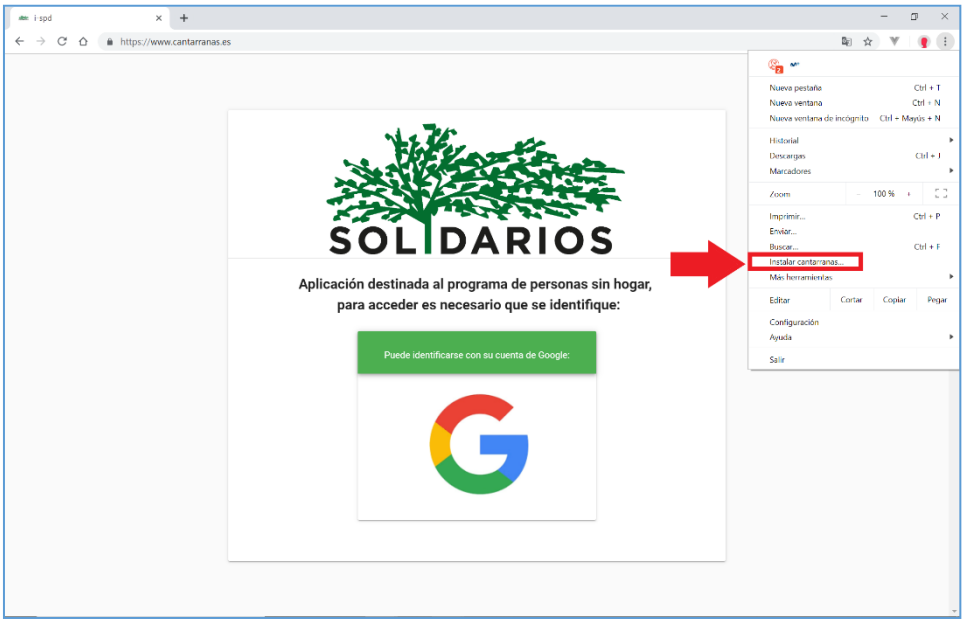
1. Abrimos la aplicación de Chrome desde nuestro ordenador:

2. A continuación, navegaremos a www.cantarranas.es (o simplemente "cantarranas.es").

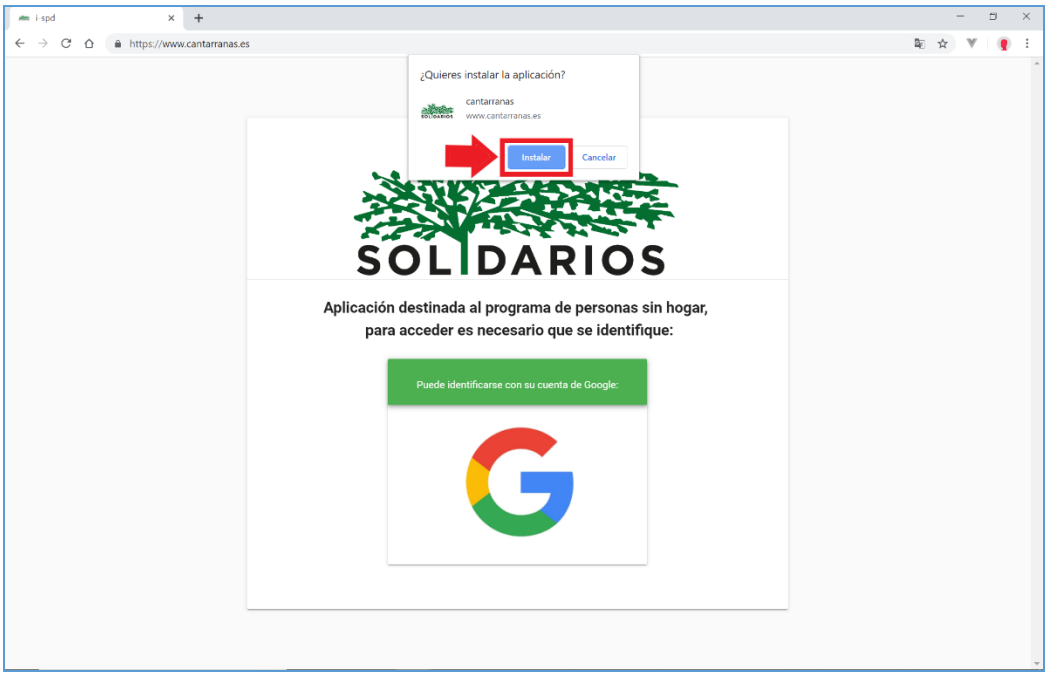
3. Una vez estemos en la Web, seleccionaremos los tres puntos que se encuentran en la esquina superior derecha de la pantalla:



4. En el menú que se nos deplegará seleccionamos la opción **"Instalar cantarranas…"**:



5. Saltará un Pop-Up donde daremos a “**Instalar**”:



6. Automáticamente se instalará la aplicación y, tras unos segundos, podremos acceder a ella a como cualquier otra aplicación de nuestro ordenador a través del siguiente icono:

B Reglas de Seguridad de Firebase

Como se menciona en la memoria del proyecto, Firebase ofrece una importante funcionalidad para que los desarrolladores que utilizan Realtime Database puedan asegurar su base de datos. Esta consiste en escribir desde las consola de Firebase las reglas de seguridad que deseas aplicar a tu base de datos y, cuando realizas un despliegue, se genera un archivo llamado *"database.rules.json"* con dichas reglas.
Cabe destacar que la propia consola de Firebase, incluye un simulador para poder probar las reglas antes de desplegar. Permite simular que la autenticación y utilizar un "uid" de usuario propio, decidir el proveedor y ejecutar la acción de escribir o leer en la parte de la base de datos a elegir y comunicando si la simulación arroja un resultado correcto o deniega el acceso.

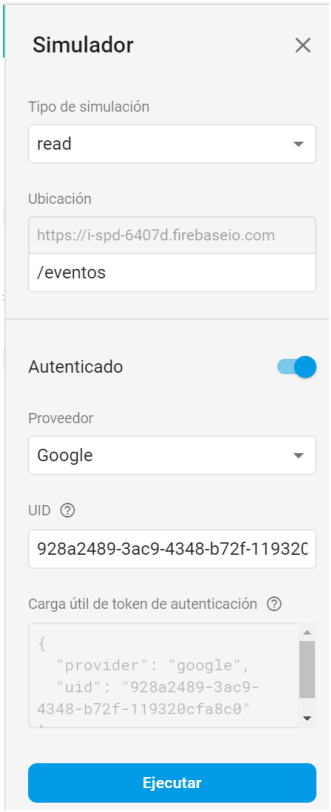


Figura 40: Simulador de Reglas de Seguridad

El desarrollo de las reglas de seguridad se ha realizado en base a la división propia de la base de datos, y se ha escrito para cada colección una la condición que se debe cumplir para que se permita lectura o escritura de manera independiente.
La condición más básica es la aplicada a la lectura de "avisos", "eventos", "peticiones", y "usuarios", consiste simplemente en que el usuario esté autenticado y su proveedor sea Google:

```
".read": "auth != null && auth.provider == 'google'",
```

Por otro lado, para poder escribir en "avisos", leer y escribir en "informes" y "posts", la condición consiste en que el usuario este dado de alta en la aplicación, ya que si esta dado de alta quiere decir que, al menos, tiene el rol de voluntario. Esto se realiza comprobando que existe un objeto en "usuarios" cuya clave sea el 'uid' del usuario autenticado:

```
".write": "root.child('usuarios/'+auth.uid).exists()"
```

Por otro lado, para poder escribir y leer “incidencias”, se requiere que el usuario autenticado tenga el rol de Coordinador o Administrador, y para escribir en “eventos” el de administrador. Para ello, buscamos el usuario autenticado con el uid y comprobamos el valor de su campo “/rol”:

```
".write": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'"
```

Por último, un caso especial es el permiso de los usuarios a escribir en la seccion de la base de datos de “usuarios”. Donde sólo pueden escribir de manera general los Administradores, que son los que dan de alta los usuarios y pueden editarlos, pero un usuario puede editar el nombre de su propio ‘usuario’, para ello se permite el acceso de escritura sólo al objeto dentro de “usuarios” cuya clave sea igual al uid del usuario autenticado, es decir sólo a su propio objeto de ‘usuario’.

```
".write": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'",
"$uid": {
  ".write": "auth != null && auth.uid == $uid"
}
```

Con todo ello, las reglas al completo son las siguientes:

```
{
  "rules": {
    "avisos": {
      ".read": "auth != null && auth.provider == 'google'",
      ".write": "root.child('usuarios/'+auth.uid).exists()"
    },
    "eventos": {
      ".read": "auth != null && auth.provider == 'google'",
      ".write": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'"
    },
    "incidencias": {
      ".read": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Coordinador' ||
          root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'",
      ".write": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Coordinador' ||
          root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'"
    },
    "informes": {
      ".read": "root.child('usuarios/'+auth.uid).exists()",
      ".write": "root.child('usuarios/'+auth.uid).exists()"
    },
    "informesTemporales": {
      ".read": "root.child('usuarios/'+auth.uid).exists()",
      ".write": "root.child('usuarios/'+auth.uid).exists()"
    },
    "peticiones": {
      ".read": "auth != null && auth.provider == 'google'",
      ".write": "auth != null && auth.provider == 'google'"
    },
    "posts": {
      ".read": "root.child('usuarios/'+auth.uid).exists()",
      ".write": "root.child('usuarios/'+auth.uid).exists()"
    },
    "usuarios": {
      ".read": "auth != null && auth.provider == 'google'",
      ".write": "root.child('usuarios/'+auth.uid+'/rol').val() == 'Administrador'",
      "$uid": {
        ".write": "auth != null && auth.uid == $uid"
      }
    }
  }
}
```

Figura 41: Reglas de Seguridad

C Fases del Proyecto e Histórico de reuniones con la ONG

Durante el desarrollo del proyecto, especialmente en la fase inicial, se han ido realizando diversas reuniones con el director del programa de personas sin hogar de Solidarios Para el Desarrollo, Jesús Sandín. A pesar de que la comunicación fue principalmente vía e-mail, se realizaron reuniones presenciales para definir requisitos y mostrar los avances. Todas ellas se han producido en las oficinas de Solidarios para el Desarrollo, en la calle Donoso Cortés de Madrid.

La primera de ellas, de carácter más informal, se produjo en Junio de 2018, con la finalidad de trasladar la idea y conocer si existía interés, con la intención de buscar el tutor adecuado para el proyecto y realizar la propuesta a la universidad en Septiembre de ese mismo año. La idea del proyecto se acogió con gran interés, especialmente se habló de sustituir el funcionamiento actual de escribir los resúmenes (Informes de Ruta) mediante grupos de Gmail, y darles una utilidad mayor. También se trasladó la importancia de poder llevar un registro de los voluntarios en activo, complementario a la base de datos de la organización. Por último se recopilaron a grandes rasgos los requisitos técnicos básicos recogidos en el apartado 3.1.2.

Previo a realizar la propuesta, se llevó a cabo un estudio de las diferentes alternativas existentes a la aplicación, recogido en el apartado "2. Estado del arte". Una vez realizada y aceptada la propuesta de TFG por parte de la universidad, se comenzó la selección de las tecnologías más adecuadas para cumplir con los requisitos definidos.

Elegidas las tecnologías, comenzó una fase de formación y desarrollo de un proyecto básico, apoyándose en la plantilla expuesta en el apartado 3.2.3. Simplemente se trataba de tener una pequeña demo que presentar y a partir de la cual definir las páginas necesarias. Esta demo consistía en una página de Bienvenida, y los formularios para escribir resúmenes e incidencias y sus respectivos muros. Por el momento se realizaba todo en local, sin utilizar Firebase.

El día 15 de noviembre de 2018, se presentó la demo a Jesús Sandín y se definieron con más detalle los requisitos funcionales:

- **Páginas y funcionalidades**
    - Página de Login, con el logo de Solidarios.
    - Página de "Bienvenida" con las novedades y avisos, editable por los Administradores.
    - Calendario de actividades con turnos de limpieza y días de cierre de Cantarranas.
    - Página con la información de la cuenta de usuario donde poder darse de baja.
    - Página con información de contacto de la ONG.
    - Acceso directo a la página web y al manual del voluntario.
    - Página donde poder escribir los resúmenes y campos necesarios en ellos.
    - Página para consultar los resúmenes con posibilidad de filtrar para consultar resúmenes de otras rutas.
    - Página para incidencias y seguimientos. Canal de comunicación directa entre Coordinadores y Administradores, donde se puedan abrir incidencias e ir modificando su estado. Y campos necesarios en las incidencias.

    - Página para Administrar usuarios, donde poder consultar, editar y eliminar usuarios.
- **Login y Usuarios.**
    - Iniciar sesión con la cuenta de Gmail. Más adelante se podría necesitar abrir a otras cuentas si resulta necesario.
    - El alta en la aplicación deberá ser aprobada por un Administrador.
    - Posibilidad de eliminar usuarios y editarlos (cambiar su rol o ruta)
    - Necesidad de dividir los usuarios por Roles que limiten el acceso a las páginas:
        - Invitado: Página de Bienvenida, Mi Cuenta, Contacto y Calendario.
        - Voluntario: Invitado + Páginas de escribir y consular resúmenes.
        - Coordinador: Voluntario + Páginas de escribir y consultar incidencias.
        - Administrador: Coordinador + Páginas de administración de usuarios y escribir novedades en la página principal.
    - Adicionalmente, se definió que cada usuario podía eliminar el contenido que publicado con su usuario en la aplicación y, los administradores podrían borrar contenido de otros usuarios.
- **Funcionalidades y aspectos extra.**
    - Se planteó la posibilidad de implantar una funcionalidad extra con la API de Google Maps o una similar, utilizando la ubicación de los dispositivos. En la que los coordinadores pudieran añadir "chinchetas" donde duermen las personas que visitamos. Teniendo un color para cada ruta, y en cada chincheta información de la hora a la que se le visita y si solapa con otra organización.
    - Se comentó la posibilidad de contactar con informáticos de la ONG, para solicitar usar el dominio de la página web, creando un subdominio para la app.
    - Además, se transmitió que si la aplicación funcionaba correctamente durante el curso 2019/2020, se estudiaría replicar para el programa de Personas Sin Hogar en otras ciudades como Sevilla o Granada.
    - También se trató la necesidad de realizar manuales de uso de la aplicación para voluntarios y administradores.
- **Fechas**
    - Se fijó comenzar a utilizar la aplicación en fase beta en el mes de Mayo, con la finalidad de encontrar y solucionar errores, así como recoger propuestas de mejora.
    - Si las pruebas fueron satisfactorias, se comenzaría a usar la aplicación de manera oficial en Septiembre, con el comienzo del curso 2019/2020, dejando de utilizar los grupos de Gmail y otras herramientas.

El 18 de Enero de 2019, se realizó la reunión de Coordinación anual, donde se reúnen los Coordinadores de todas las rutas para comentar cómo va el curso y ver qué cosas se pueden mejorar. En dicha reunión, se aprovechó para contar el proyecto en marcha de la aplicación, la acogida fue realmente positiva y los Coordinadores insistieron en los siguientes puntos

- Que la aplicación ocupara el mínimo de memoria posible.
- Poder escribir los resúmenes desde ahí y poder consultar resúmenes de otras rutas.

Adicionalmente, propusieron funcionalidades nuevas:

- Una sección común a todos los voluntarios donde poder compartir cosas, como libros leídos u organizar quedadas, similar a una página de Facebook.
- Posibilidad de que al abrir una incidencia se puedan ir haciendo comentarios sobre ella, similar a un hilo en un foro.

Durante los siguientes meses, se estuvo desarrollando la aplicación a nivel Frontend y también Backend, configurando Firebase y realizando los primeros despliegues en el hosting de Firebase.

Una vez se tuvo la aplicación más avanzada, se realizó una nueva reunión con Jesús Sandín, para realizar una nueva demo, cuyo resultado fue muy satisfactorio y se perfilaron algunos detalles:

- Implementar una forma simplificada y rápida de crear los eventos específicos de los turnos de limpieza en el Calendario.
- Incluir la información en la página de contacto de las sesiones informativas que se realizan los lunes, con un enlace para poder inscribirse. Así como el teléfono de contacto que fue facilitado en la reunión.
- Dividir el campo de las personas visitadas de los informes de ruta por género. Por un requerimiento de la Comunidad de Madrid.
- Añadir la posibilidad de guardar el informe de ruta mientras se escribe, para poder continuar con él, completarlo y mandarlo. Con el fin de que los voluntarios que lo deseen puedan ir realizando el informe a medida que hacen la ruta sin perder el progreso.
- Se habló de la necesidad de tener un apartado de Política de Privacidad, términos y condiciones de la aplicación, que fuera necesario leer previamente a enviar la solicitud de alta. Se realizó un borrador con ayuda de un técnico de la ONG, y cuando se completó se confirmó vía e-mail.
- Se facilitó el logo oficial de Solidarios. Que se añadió en distintas partes de la aplicación y se adaptó para ser el icono de la misma.
- Por último, se definió el nombre de la aplicación como “Cantarranas”, al igual que el local donde se reúnen los voluntarios a recoger alimentos antes de salir de ruta.

La comunicación continuó vía e-mail, a medida que iba avanzando el desarrollo del proyecto, se fue fijando la fecha de comienzo de las pruebas, que finalmente fue el 20 de Mayo. Sus resultados pueden verse en el apartado “5.2. Pruebas reales” del presente documento.

Con las pruebas, se continuó puliendo la aplicación y, durante los meses de Julio y Agosto, se espera avanzar en los puntos recogidos en el apartado “6.2. Trabajo futuro”, para que en septiembre la aplicación entre en la fase de uso real lo más completa y funcional posible.

Por último, el día 13 de junio se realizó la reunión anual de cierre de curso del programa, donde asisten los coordinadores de las diferentes rutas y los responsables del programa. Uno de los puntos de dicha reunión, fue la valoración de la aplicación, resultando muy positiva y confirmando que se comenzará a utilizar de manera completa con el comienzo del nuevo curso en septiembre de 2019.